AVT electronics



HANDLEIDING

VOOR DE

GoldStar / YASHICA

MSX
COMPUTERS

A.C.J. GROENEVELD

HANDLEIDING
voor uw
MSX
computer

HANDLEIDING voor uw MSX computer

A. C. J. Groeneveld

uitgeverij STARK - TEXEL
postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

ISBN 90 6398 394 8

INHOUD

MSX-SLEUTELWOORDEN OP ALFABETISCHE VOLGORDE

1 INLEIDING

Gefeliciteerd met de aanschaf van uw MSX computer. Met deze aanschaf verzekerde u zich van een computer die voldoet aan de MSX standaard, een waarborg voor betrouwbaarheid, uitwisselbaarheid en een keur van beschikbare programmatuur op allerlei gebied.

Wat betekent MSX eigenlijk? MSX is een afkorting van MicroSoft eXtented basic. Met de merknaam MSX heeft MICROSOFT, het toonaangevende software huis op het gebied van microcomputers, een zéér belangrijke en bitter noodzakelijke stap gedaan in de richting van de standaardisering van hobby- en microcomputers en hun taal. Eindelijk is het mogelijk om programmatuur aan te schaffen die op vele merken micro's werkt; een behoefte die al jarenlang bestond.

MSX is meer dan alleen een standaard BASIC taal. MSX is een volledig standaard concept voor microcomputers. Niet alleen de taal maar ook de opbouw van het toetsenbord is door MSX standaard voorgeschreven voor de processor, de geluidsgenerator, de beeldschermprocessor etcetera. Ook de geheugenindeling ligt vast. Hierdoor wordt de mogelijkheid geschapen om behalve programma's ook randapparatuur (ROM-casettes, floppy disk eenheden, joy sticks e.d.) tussen de verschillende merken microcomputers te kunnen uitwisselen.

Deze handleiding is een vereenvoudigde en verkorte uitgave van het boek
MSX BASIC HANDBOEK VOOR IEDEREEN
uitgegeven door uitgeverij Stark-Texel en te bestellen onder ISBNnummer 90 6398 100 7.
Met deze handleiding die speciaal voor uw computer werd samengesteld, kunt u met een veilig gevoel uw eerste schreden in de wereld van het MSX-BASIC zetten. Later, wanneer u alles van uw computer wilt weten, is het raadzaam om het uitgebreide handboek aan te schaffen.

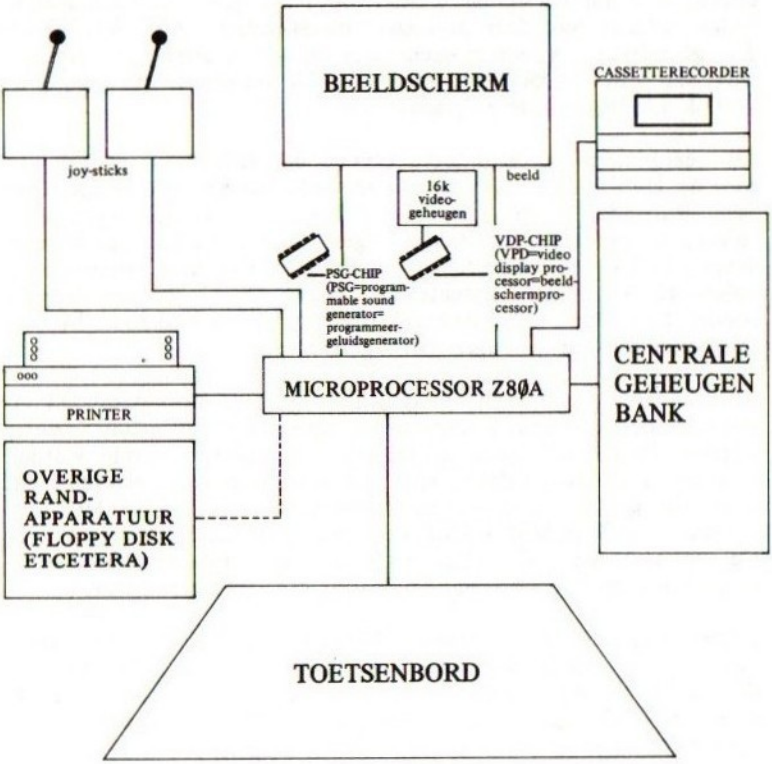
Veel genoeglijke uren toegewenst met uw computerhobby.

januari 1985,
A.C.J. Groeneveld.

Zoals in de inleiding reeds werd opgemerkt, beperkt de MSX standaard zich niet alleen tot de BASIC taal. Ook het hardware-concept wordt duidelijk door deze standaard bepaald.

2.1 De hardware van een MSX-computer

Hieronder volgt een schema van de algemene opbouw van een MSX-computer.



Als standaard microprocessor is binnen de MSX-standaard gekozen voor de Z-80-A microprocessor (of een processor met gelijke specificaties). De Z-80 is de meest verbreide microprocessor en vormt al jaren het hart van vele microcomputers. Alhoewel de Z-80 een 8 bits microprocessor is en door velen als ouderwets wordt betiteld ten opzichte van de 16 bits microprocessoren, voldoet deze chip nog steeds uitstekend op hobby- en semi-professioneel gebied. De verwachting is, dat in de toekomst nog vele nieuwe computermodellen zullen worden ontworpen op basis van de Z-80 microprocessor.

Als geluidsgenerator is gekozen voor de AY-3-8912 geluidsprocessor (of een processor met gelijke specificaties). Deze processor is een computer op zich en regelt alle geluidseffecten van de MSX-computer. Ook voor deze geluidsgenerator geldt dat het een oude bekende is binnen de computerwereld. Echter, door zijn grote aantal mogelijkheden voldoet ook deze processor uitstekend. De AY-3-8912 bezit drie geluidskanalen, één ruisgenerator en een variëteit van effecten. Hierdoor is het bijvoorbeeld goed mogelijk om een driestemmig stuk muziek met slagwerk te programmeren.

Als beeldschermprocessor werd binnen de MSX-standaard gekozen voor de TMS 9929A processor (of een processor met gelijke specificaties). Bijzonder is dat deze zeer geavanceerde grafische processor volgens de standaard dient te zijn uitgerust met een apart stuk werkgeheugen (RAM) van maar liefst 16 kilobytes. Hierdoor kan een hoog oplossend vermogen (fijngetekend beeldscherm) worden bereikt zonder dat meteen een aanzienlijk deel van het toch al zo snel tekort schietende centrale geheugen wordt gereserveerd.

Als standaard magnetisch opslagmedium gaat de MSX-standaard uit van een doodgewone cassetterecorder. Op een normale geluidscassette worden programma's gezet en kunnen ook andere gegevens worden geplaatst. Bijzonder is dat de MSX-standaard uitgaat van een relais dat de recordermotor aan en uit kan schakelen. Dit relais is via het MOTOR-commando ook in MSX-BASIC te besturen; de handige programmeur kan op deze wijze dit relais ook voor andere besturingsdoeleinden gebruiken zoals bijvoorbeeld de besturing van een diaprojector.

Uiteraard voorziet de standaard in de aansluiting van een printer. Deze aansluiting geschiedt via het standaard Centronics parallelle protocol. De printer kan een afdrukeenheid zijn met de mogelijkheid om de standaard MSX grafische symbolen af te drukken maar mag ook een algemene printer zonder deze mogelijkheden zijn.

De joy-sticks vormen onmisbare attributen voor de spelletjes-fanaat! Als zodanig ontbreken deze niet in de MSX-standaard. Programmeurs die spelletjes in MSX-basic willen schrijven, vinden een keur van eenvoudige commando's, waarmee de joy-sticks volledig kunnen worden uitgebuit.

Alhoewel MSX in versie 1.0 nog geen verdere randapparatuur ondersteunt, is aansluiting van overige randapparatuur, met name van floppystations, ook mogelijk. Een speciale BASIC-uitbreiding is noodzakelijk om deze randapparatuur aan te kunnen spreken (het MSX-disk basic).

Als beeldscherm kan natuurlijk een mooie en dure kleurenmonitor worden aangesloten, maar een eenvoudige draagbare zwart-wit televisie werkt ook al zeer bevredigend. MSX schrijft zowel een composiet-uitgang (naar de antenne-ingang van het televisietoestel) als een video-uitgang met apart geluidskanaal (voor monitors) voor.

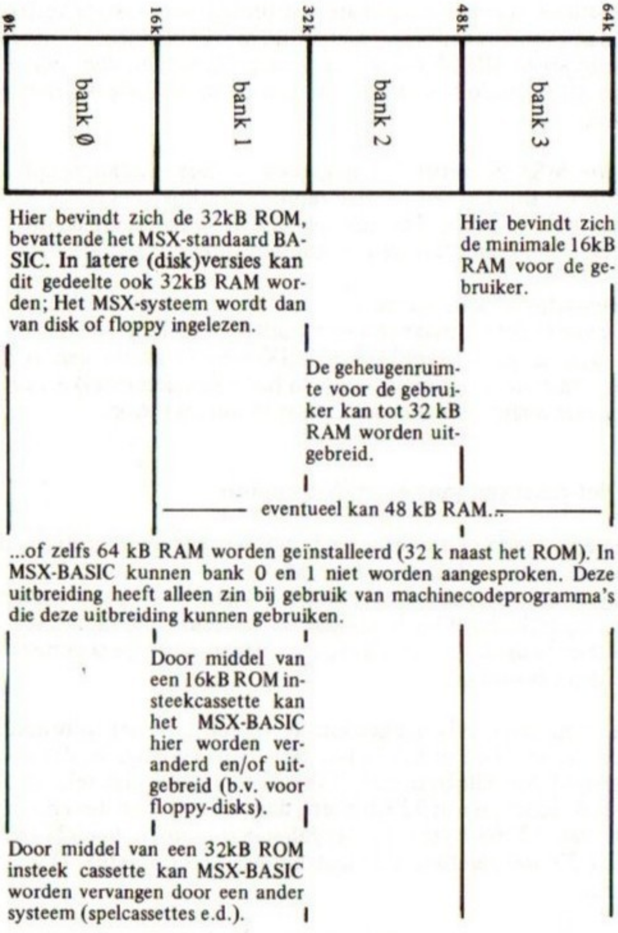
2.2 Het geheugen van een MSX-computer

Het geheugen van een computer heeft een bepaalde grootte, die men uitdrukt in bytes, kilobytes of megabytes.
Een byte is een geheugen-eenheid waarin één enkel karakter kan worden opgeslagen. Om bijvoorbeeld de tekst "MSX-BASIC" in het computergeheugen op te slaan, zijn negen geheugeneenheden ofwel negen bytes benodigd.

Indien men over geheugengrootten spreekt, is het gebruikelijk om deze uit te drukken in kilobytes. Het merkwaardige is, dat in de computerwereld een kilobyte niet 1000, maar 1024 bytes telt. Indien men het over 32 kilobytes of 32 Kb heeft, dan bedoelt men dus een geheugengrootte van 32768 bytes. In dat geheugen kunnen dus 32768 lettertekens (of 32768 machinecode-instructies, zie hoofdstuk 4) worden opgeslagen.

Een megabyte telt weer 1024 kilobytes. Binnen de microcomputerwereld heeft men het meestal pas over megabytes, indien men het over de opslagcapaciteit op magneetschijf of diskette heeft.

De centrale geheugenbank van de MSX-computer bestaat in werkelijkheid uit **vier** geheugenbanken van elk 16 kilobytes. In het volgende schema wordt de mogelijke bezetting van deze geheugenbanken beschreven.

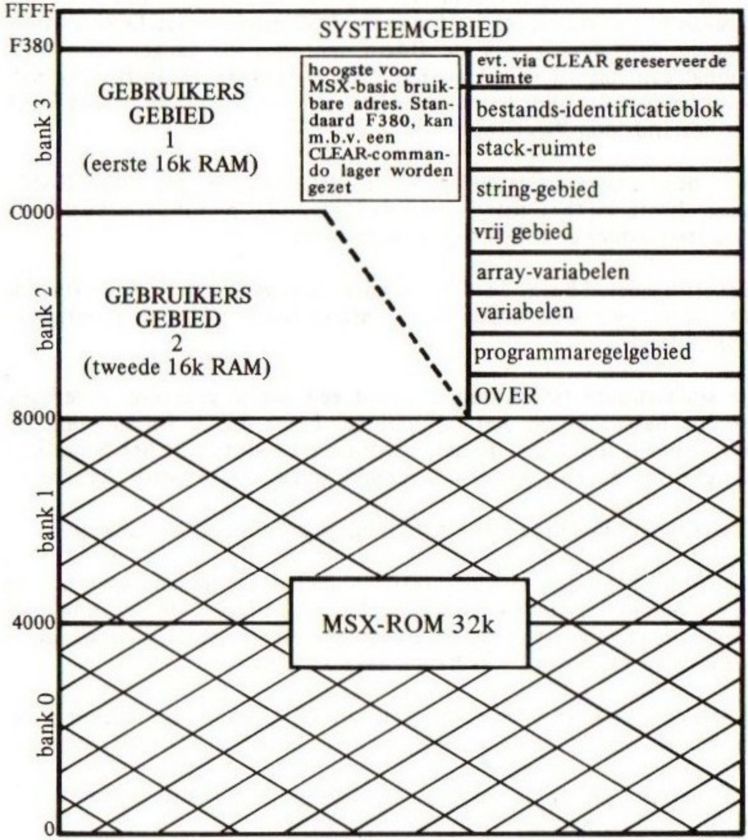


In het meest eenvoudige systeem schrijft de MSX-standaard 32 kilobytes ROM (READ ONLY MEMORY=alleen uitleesbaar geheugen met vaste inhoud) voor waarin het MSX-systeem dient te zijn gecodeerd. Daarbij dient er minimaal 16 kilobytes aan RAM (RANDOM ACCESS MEMORY=algemeen toegankelijk en veranderbaar geheugen) in bank 3 aanwezig te zijn. Door middel van uitbreidingen of ROM-cassettes kan de totale geheugenopbouw worden veranderd.

2.3 MSX-BASIC en het RAM

Het MSX-BASIC gebruikt het beschikbare RAM op een wijze die is toegelicht in het volgende schema.

SCHEMA GEHEUGENBANK MSX-BASIC



Het allerbovenste gedeelte is altijd gereserveerd voor het MSX systeem. Dit gedeelte dient als kladblaadje voor interne gegevens, dat het systeem zo nu en dan nodig heeft.

Onder dit systeemgedeelte volgen diverse blokken die elk moment, afhankelijk van het werkende programma, anders van grootte kunnen zijn.

Van boven naar beneden komen we als volgende blok de via CLEAR gereserveerde ruimte tegen. Deze ruimte kan binnen de BASIC-taal worden gecreëerd via het CLEAR sleutelwoord. Deze ruimte kan door de gevorderde programmeur vervolgens worden gebruikt voor het opslaan van routines, geschreven in de Z-80 machinetaal. Deze routines kunnen dan weer vanuit de BASIC-taal worden aangeroepen door middel van het DEFUSR en het USR sleutelwoord. In het normale geval bestaat dit blok niet; onder het systeem volgt dan onmiddellijk het bestandsidentificatieblok.

Het bestandsidentificatieblok bevat gegevens die het MSX-BASIC nodig heeft bij het uitwisselen van gegevens met bijvoorbeeld een cassetterecorder of een floppy disk eenheid.

Het stringgebied bevat de alfanumerieke gegevens (gegevens, bestaande uit letters, cijfers en leestekens) waarmee het betreffende programma manipuleert.

De stack-ruimte (stack=stapel) bevat een aantal gegevens in verband met de besturing van het programma. Later, bij de behandeling van bijvoorbeeld het GOSUB sleutelwoord zal blijken dat het benodigde terugkeeradres een gegeven is dat in deze stack-ruimte wordt opgeslagen.

Onder de stack-ruimte volgt een klein, ongebruikt stukje geheugen.

Hieronder bevinden zich de array-variabelen. De namen, specificaties en numerieke inhoud liggen in dit gebied opgeslagen. De alfanumerieke inhoud van de array-variabelen ligt echter in het stringgebied opgeslagen. Voor de betekenis van variabelen: zie hoofdstuk 6.

Onder de array-variabelen bevinden zich de niet-array-variabelen. Deze variabelen zijn op een dergelijke manier binnen dit gebied opgeslagen.

Als laatste blok volgt het blok met programmaregels. Het feitelijke BASIC-programma ligt in dit blok opgeslagen.

Indien er nog geheugen over is, ligt dat onder het programmaregelgebied.

2.4 Het toetsenbord van een MSX-computer

Zoals bij de meeste microcomputers het geval is, bezit ook de MSX-computer een toetsenbord met een standaard indeling. In Nederland is het zogenaamde QWERTY-toetsenbord het meest gebruikt.

Het toetsenbord valt onder te verdelen in drie soorten toetsen, te weten de karaktertoetsen, de controletoetsen en de functietoetsen.

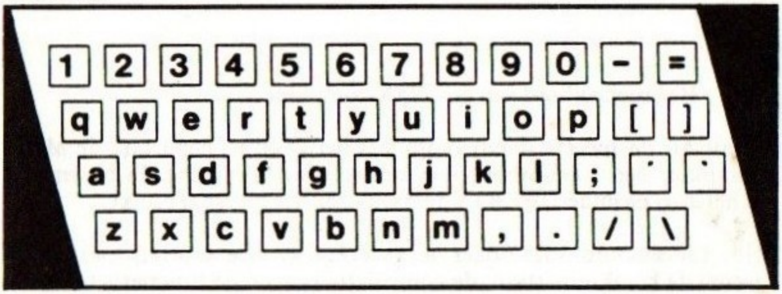
De karaktertoetsen genereren bij intoetsing de vermelde letters en cijfers; op het eerste gezicht niets bijzonders dus. Echter, in combinatie met de functietoetsen SHIFT, CODE en GRAPH zijn er veel meer tekens te genereren dan op het eerste gezicht duidelijk is. Om deze tekens uit te proberen behoeft men slechts de computer aan te zetten en te wachten op de copyrightmelding, gevolgd door de Ok-melding. Hierna kunnen en mogen alle toetsaanslagen worden uitgeprobeerd. Let u in eerste instantie niet op de vele foutmeldingen die de MSX-computer kan geven als protest op uw door de computer niet begrepen intoetsingen.

De zes schema's op de volgende twee pagina's geven de karakters weer die via het toetsenbord kunnen worden gegenereerd.

Buiten deze uitgebreide set van karakters kunnen ook acties aan de MSX-computer worden ontlokt. Dit gebeurt via de zogenaamde controletoetsen. De MSX-standaard vereist de aanwezigheid van de volgende controletoetsen:

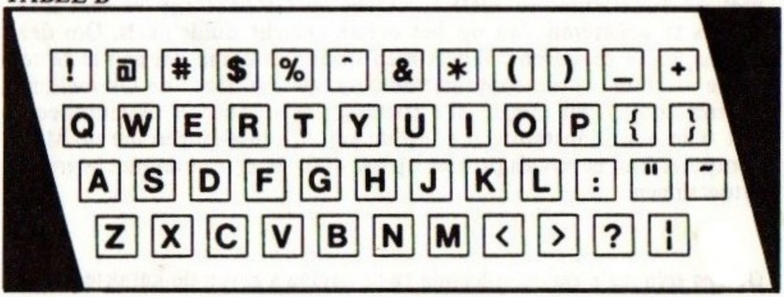
toets	functie
ESC	Kan in programma's worden gebruikt, maar heeft binnen het MSX-basic geen functie.
TAB	Met deze toets kan direct naar een volgende kolom op het beeldscherm worden gesprongen. Heeft over het algemeen niet veel nut. Het beeldscherm is standaard in kolommen van 14 tekens ingedeeld.

TABEL A



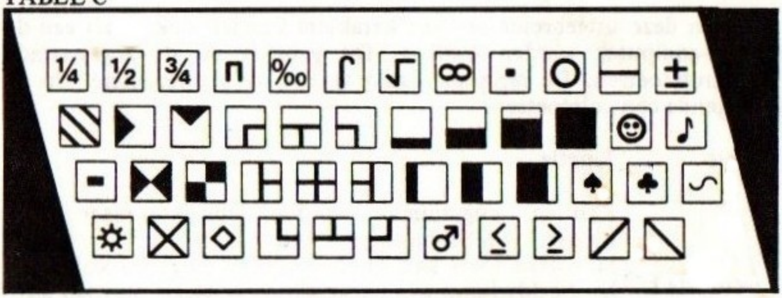
Karakters die normaal ontstaan bij het intoetsen.

TABEL B



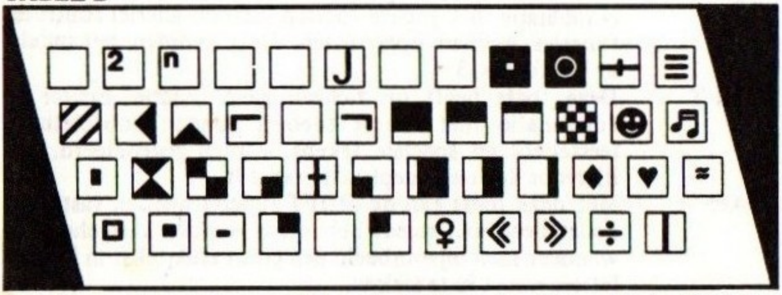
Karakters die ontstaan indien de toetsen tegelijk met de SHIFT-toets worden ingedrukt.

TABEL C



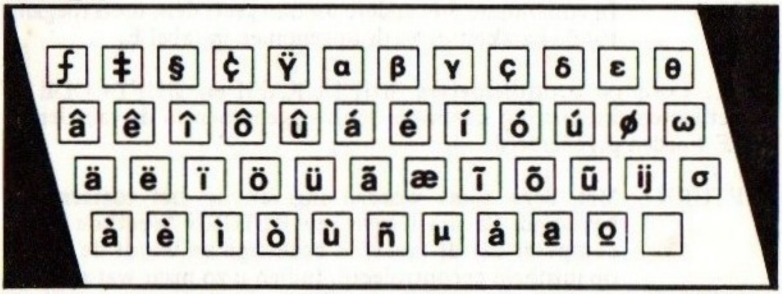
Karakters die ontstaan indien de toetsen tegelijk met de GRAFH-toets worden ingedrukt.

TABEL D



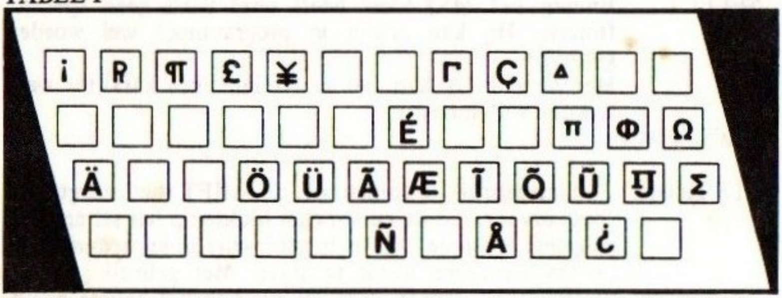
Karakters die ontstaan indien de toetsen tegelijk met de SHIFT- en GRAFH-toets worden ingedrukt.

TABEL E



Karakters die ontstaan indien de toetsen tegelijk met de CODE-toets worden ingedrukt.

TABEL F



Karakters die ontstaan indien de toetsen tegelijk met de SHIFT- en CODE-toets worden ingedrukt.

CTRL	Deze toets heeft op zichzelf geen functie. Echter in combinatie met andere toetsen kunnen allerlei controlefuncties worden opgeroepen. Deze worden behandeld in hoofdstuk 3.
SHIFT	Deze toets heeft op zichzelf geen functie. Echter in combinatie met andere toetsen kunnen hoofdletters, leestekens en speciale tekens worden gegenereerd. Zie hiervoor de eerder geplaatste tabel B.
CAPS	Met deze toets kan de SHIFT functie worden vastgezet zodat deze niet steeds behoeft te worden vastgehouden wanneer men bijvoorbeeld een groot stuk tekst in hoofdletters wenst in te tikken.
GRAPH	In combinatie met andere toetsen geeft deze toets toegang tot een extra set karakters zoals eerder in tabel C opgenomen.
CODE	In combinatie met andere toetsen geeft deze toets toegang tot de karakterset zoals opgenomen in tabel E.

N.B.: Door de controletoetsen SHIFT-GRAPH en SHIFT-CODE te combineren, kunnen de karakters zoals opgenomen in de tabellen D en F worden gegenereerd.

RETURN	Met deze toets worden ingaven op het toetsenbord bevestigd. Pas na ingave van RETURN 'weet' de MSX-computer dat de ingave is beëindigd en wordt de ingave op juistheid gecontroleerd. Indien u zo maar wat met het toetsenbord speelt, bijvoorbeeld om de karakterset uit te proberen, merkt u dat de computer pas na ingave van de RETURN-toets een eventuele foutmelding geeft.
SELECT	Binnen het MSX-basic heeft deze toets geen speciale functie. Hij kan echter in programma's wel worden gebruikt.
BS of BACKSPACE	Met deze toets kan het laatst ingetikte karakter weer worden verwijderd.
CLS/HOME	Zonder gebruik te zamen met de SHIFT toets zorgt deze toets ervoor, dat de cursor (het blokje op het scherm dat aangeeft waar de ingave is gebleven) links boven in de beeldschermhoek komt te staan. Met gebruik van de SHIFT toets wordt daarbij ook nog het gehele beeldscherm uitgewist.
INS	Met behulp van deze toets kan in een bestaande regel een

	karakter of een stuk tekst worden tussengevoegd. Door een intoetsing wordt het tussenvoegen geactiveerd; door nog een intoetsing of door gebruik van één van de pijltoetsen wordt het tussenvoegen weer gedeactiveerd.
DEL	Met behulp van deze toets kan één karakter worden verwijderd op elke plaats binnen een regel. De rest van de regel wordt dan aangeschoven.
STOP	Met deze toets kan een werkend BASIC-programma worden stilgezet. Een volgende intoetsing zorgt ervoor dat het betreffende programma dan weer verder gaat. In combinatie met de CTRL-toets kan met deze toets een programma worden onderbroken.
PIJLTOETSEN	Met deze toetsen kan de cursor in horizontale en vertikale richting worden verplaatst. Door intoetsing van twee toetsen tegelijk kan de cursor ook diagonaal (schuin) worden verplaatst. De pijltoetsten kunnen in een BASIC-programma te zamen als joy-stick dienst doen.

Alle functietoetsen zijn gekenmerkt door een afgekorte, Engelstalige naam. Ter verduidelijking volgt voor elke toets hier een verklaring van de naam:

ESC	ESCape=ontsnappen
TAB	TABulation=tabulatie (kolomindeling)
CTRL	ConTRoL=controle
SHIFT	SHIFT=opschuiven
CAPS	CAPitalS=hoofdletters
GRAPH	GRAPHics=grafische tekens
CODE	CODE=codering
RETURN	RETURN=terugkeren (naar de linkerkant van het beeldscherm, één regel lager)
SELECT	SELECT=selecteren, uitzoeken
BACKSPACE	BACKSPACE='terugspatie' een spatieteken wordt de 'verkeerde' kant uit over het foute karakter gezet.
CLS	CLear Screen=beeldscherm schoonmaken
HOME	HOME=huis. De cursor wordt naar 'huis' teruggestuurd.
INS	INSert=tussenvoegen
DEL	DELete=verwijderen
STOP	STOP=stoppen, stilstaan

De functietoetsen zijn genummerd van 1 tot en met 10. 'Onder' deze

functietoetsen zijn vaste teksten opgenomen die veelvoudig terugkeren. Door middel van de intoetsing van de betreffende functietoets wordt dan de gehele tekst op beeldscherm geplaatst. De functietoetsen kunnen via het MSX-basic eventueel van functie worden veranderd. De functies worden (gedeeltelijk) onder in het beeld zichtbaar gemaakt. Deze regel is eventueel uitschakelbaar; ook dit kan in MSX-basic geschieden.

Normaal toont het beeldscherm de eerste vijf functies, corresponderend met functietoets 1 tot en met 5. Door alléén de SHIFT in te drukken, worden de functies, corresponderend met functietoets 6 tot en met 10, onder in beeld afgedrukt.

De MSX-standaard voorziet in een full screen editor. Een editor is de Engelse naam voor een tekstverzorgend programma. Full screen wil zeggen dat tekst over het gehele beeld kan worden verzorgd.

Om wat met de MSX full screen editor te oefenen, is nog geen enkele kennis van het MSX-basic noodzakelijk. Slechts enkele gegevens zijn in dit stadium belangrijk:

1) Begin een regel tekst altijd eerst met een regelnummer dat groter is dan of gelijk is aan 0. Het regelnummer moet kleiner dan 65530 blijven.
2) Een ingetikte regel mag groter zijn dan de breedte van een beeldschermregel, maar moet kleiner blijven dan 256 tekens.
3) Een regel mag voorlopig elke volgorde van karakters bevatten.
4) Een regel dient altijd te worden beëindigd door de RETURN-toets.
5) Een op een of andere wijze veranderde regel wordt pas definitief opgenomen nadat een RETURN is ingegeven.

Een geldige tekst om in de MSX-editor mee te oefenen, is bijvoorbeeld:

```
1 IK OEFEN WAT MET DE MSX EDITOR
10 HDJS JSJS HSHSHSHS TETETE
123 IK TIK ZOMAAR WAT IN ...
```

maar elke zelf bedachte (nonsens-) tekst is uitstekend.

3.1 LIST

Nadat u een aantal regels op deze wijze heeft ingetikt, kunt u deze regels op volgorde van regelnummer weer opvragen door middel van het LIST-commando. Tikt u het woord LIST (hoofdletters of kleine letters maken hier geen verschil) in, gevolgd door de RETURN-toets. In een vaartje ziet u de regels keurig en in volgorde over het beeldscherm gaan. Maakt u zich niet ongerust over verdwijnende regels op het beeldscherm; deze zijn geregistreerd en kunnen door het list-commando weer worden opgeroepen.
Indien u slechts één regel wilt opvragen, tikt u dan LIST in, gevolgd door het betreffende regelnummer en een RETURN-toets. Een beperkte

hoeveelheid tekst kan worden opgeroepen door het LIST-commando in te toetsen, gevolgd door een eerste en een laatste regelnummer, door middel van een min-teken van elkaar gescheiden.
LIST biedt nog meer mogelijkheden, maar die zijn in het kader van dit hoofdstuk niet belangrijk.

3.2 De pijltoetsen, NEW

Op de boven beschreven wijze kunt u het beeldscherm naar believen met tekst vullen. De cursor, het blokje op het beeldscherm dat aangeeft waar u met intikken gebleven bent, loopt steeds keurig met uw ingave mee.
De cursor kunt u met behulp van de pijltoetsen in de aangegeven richtingen verplaatsen. Zo kunt u de cursor bijvoorbeeld midden in een stuk tekst zetten. Als u daarna wat letters intikt, dan komen die in de plaats van de lettertekens die eerder op die plaats stonden. Wanneer u na de verandering een RETURN-toets ingeeft, dan merkt u dat de cursor aan het begin van de volgende regel gaat staan. De door u aangebrachte wijziging is opgenomen in het programmageheugen. U kunt dit met het LIST-commando natuurlijk nog even controleren.

De pijltoetsen maken het mogelijk om op willekeurige plaatsen op het scherm wijzigingen aan te brengen en deze vervolgens permanent te maken.

Een andere mogelijkheid is het copiëren van regels. Geef bijvoorbeeld eens de tekst NEW in, gevolgd door een RETURN. Controleer daarna met een list of alle regels daadwerkelijk door de computer 'vergeten' zijn (NEW betekent NIEUW).
Tik hierna in (en vergeet de RETURN niet):

```
10 DIT IS DE EERSTE REGEL VAN VELE
```

Ga vervolgens met de pijltoetsen naar het regelnummer 10 en maak hier een regelnummer 20 van door alleen de 1 in een 2 te veranderen. Geef vervolgens (uiteraard) een RETURN in en doe daarna een LIST. U ziet dat de regel nu twee maal is opgenomen, eenmaal onder regelnummer 10 en eenmaal onder regelnummer 20.
Nu kan regel 20 bijvoorbeeld worden aangepast door met de pijltoetsen naar de juiste plaats te gaan en de tekst EERSTE te vervangen door de tekst TWEEDE. Na RETURN is ook deze verandering weer definitief.

Indien een regel niet meer is gewenst, kan deze worden verwijderd door alleen het regelnummer in te geven, gevolgd door een RETURN. Ook deze verwijdering kunt u natuurlijk weer controleren met de LIST-functie.

3.3 Verwijderen van tekens

Laten we nog eens een NEW-commando ingeven (gevolgd door de RETURN) en vervolgens met een SHIFT-CLS (of een CTRL-L) het beeldscherm schoonmaken. We beginnen met een schone lei. Als eerste regel tikken we in:

```
1 DIT IS DAN DE EERSTE REGEL
```

Nu gaan we om te oefenen het woordje DAN uit deze eerste regel verwijderen. Dit doen we door met de pijltoetsen naar de letter D van DAN te gaan en vervolgens vier keer de DEL-toets in te drukken. De DEL-toets dient vier keer te worden ingedrukt omdat ook het spatieteken na DAN moet worden verwijderd.

Op deze wijze kunnen stukken tekst worden verwijderd uit een regel. Na de RETURN is de verkorte regel definitief opgenomen; met een LIST kan dat worden gecontroleerd.

3.4 Invoegen van tekens

Het kan natuurlijk ook zijn dat we juist een stuk tekst willen tussenvoegen in een bestaande regel. Als oefening kunnen we bijvoorbeeld proberen om het woordje DAN weer in de regel uit het laatste voorbeeld terug te zetten.

Hiervoor zetten we met de pijltoetsen de cursor op de letter D van het woordje DE. Tik vervolgens de INS-toets in; de cursor verandert nu in een streepje.
Tik nu het woordje DAN in en vergeet de spatie na DAN niet. Geef vervolgens een RETURN om de regel definitief vast te leggen en controleer of de uitgebreide regel daadwerkelijk is opgenomen met een LIST.

Het zal blijken dat het woordje DAN netjes is tussengevoegd. Pas bij het invoegen op de volgende punten:

1) door de INS-toets nog een keer in te drukken, wordt de invoegfunctie weer opgeheven en wordt het volgende karakter weer gewoon over het oude karakter heen geplaatst.
2) een pijltoets heft de invoegfunctie eveneens op.

3.5 De overige controletoetsen

Het zal duidelijk zijn dat met de MSX-editor het intikken van programmaregels alsmede het corrigeren van deze regels een eenvoudig karwei is. Met een beperkt aantal duidelijke controletoetsen zijn vele mogelijkheden te verwezenlijken, onder andere:

- het intoetsen van een stuk tekst
- het overschrijven van een stuk tekst
- het verwijderen van een stuk tekst
- het tussenvoegen van een stuk tekst.

Behalve de nu bekende controletoetsen (RETURN, INS, DEL en de pijltoetsen) zijn er nog meer functies op te roepen die weleens gemakkelijk zijn. Eén kwamen we er al tegen: de BACKSPACE (BS) voor het corrigeren van het laatste karakter.

De overige functies laten zich slechts oproepen door middel van het intoetsen van de CTRL-toets te zamen met een ander karakter. Eén zo'n functie kwamen we reeds tegen: de CTRL-L (druk tegelijk de CTRL en de L-toets in) voor het schoonmaken van het beeld.

Hieronder volgt een schema met de overige functies van de editor. Om te oefenen kunt u het beste eerst een flink stukje tekst intikken en vervolgens één voor één de functies uitproberen.

CONTROLEFUNCTIES VOOR DE MSX-EDITOR

toets die te zamen met de CTRL-toets moet worden ingetoetst	functie van deze toets
B	de cursor gaat terug naar het begin van het vorige woord.
C	onderbreken van de ingave zonder dat de regel definitief wordt opgenomen.

E	alle tekst op de betreffende regel, rechts van de de cursor, wordt verwijderd.
F	de cursor gaat verder naar het begin van een volgend woord.
G	een kort geluidssignaal is hoorbaar, verder geen functie.
H	heeft dezelfde functie als BACKSPACE (BS).
I	heeft dezelfde functie als de TAB; de cursor verplaatst zich naar de volgende kolom. Het beeldscherm is standaard in kolommen van 14 tekens ingedeeld.
J	de cursor gaat naar de volgende regel. Eventueel wordt de beeldscherminhoud één regel opgeschoven (gebeurt niet met de pijl-naar-beneden-toets).
K	heeft dezelfde functie als HOME.
L	heeft dezelfde functie als CLS.
M	heeft dezelfde functie als RETURN.
N	de cursor gaat helemaal naar het einde van de regel.
R	heeft dezelfde functie als INS.
U	de volledige regel wordt schoongemaakt.
X	heeft dezelfde functie als SELECT (geen).
\	heeft dezelfde functie als pijl naar rechts.
]	heeft dezelfde functie als pijl naar links.
^	heeft dezelfde functie als pijl naar boven.
_	heeft dezelfde functie als pijl omlaag.
DEL	heeft dezelfde functie als DEL zonder CTRL.

Pas op: hoegenaamd alle controlefuncties hebben tot gevolg, dat een eventueel geactiveerde tussenvoeging wordt gede-activeerd. Dit kan men zien aan de cursor, die dan weer de vorm van een blokje (in plaats van een streepje) krijgt.

Het is raadzaam om ervoor te zorgen, enige ervaring met de MSX-editor te verkrijgen alvorens de MSX-studie te vervolgen.

4 HET MSX-BASIC

4.1 BASIC

De naam van de taal BASIC is een afkorting van Beginners All-purpose Symbolic Instruction Code. Zoals de naam doet vermoeden, werd de taal BASIC in eerste instantie ontworpen voor beginners op het gebied van de computerwereld. Sinds de introductie van BASIC is er door diverse software-ontwerpers een keur van zogenaamde dialecten ontworpen zodat het nu niet meer mogelijk is om van een standaard BASIC te spreken. De tegenwoordige BASIC-dialecten zijn vaak zeer uitgebreid en lijken niet meer op het eerste, eenvoudige BASIC waar alle, nu meer dan 300 verschillende, dialecten van af stammen.

Het MSX-basic is één van de vele BASIC-dialecten. Echter door de bijzondere kwaliteit van dit BASIC en de ruime mate waarmee het door diverse computerfabrikanten wordt toegepast, kan het ondertussen wel de standaard BASIC voor micro-computers worden genoemd.

Het MSX-basic bevat meer dan 150 sleutelwoorden (het eerste BASIC bevatte slechts een tiental sleutelwoorden...). Met deze sleutelwoorden kunnen commando's worden gevormd waarmee de MSX-computer wordt opgedragen om te rekenen, te tekenen, geluid te maken, tekst af te drukken etcetera.

MSX betekent: MicroSoft eXtended basic ofwel 'het uitgebreide basic van Microsoft'. Uit de naamgeving blijkt wel dat de taal niet nieuw is; het normale Microsoft basic is al jarenlang de meest bekende BASIC op de wat professionelere micro-computer.

Echter, met de ontwikkeling van de kleine microcomputer, de zogenaamde huiscomputer of home-computer, groeide de noodzaak om dit BASIC aan te passen aan de moderne mogelijkheden en toe te passen op de kleine microcomputers. Uit deze noodzaak ontstond het uitgebreide Microsoft basic ofwel het MSX-basic; een BASIC waarin ervaring van vele jaren is verwerkt en waarin de meest moderne mogelijkheden zijn ondervangen.

4.2 Commando's

Zoals reeds werd opgemerkt, heeft MSX-basic ongeveer 150 sleutelwoorden waarmee commando's kunnen worden geformuleerd. Indien de formulering van deze commando's juist geschiedt, zal de computer u perfekt gehoorzamen. Indien echter een klein foutje in deze formulering zit, zal de computer u niet begrijpen en reageren met een foutmelding.

Om alvast een beetje de smaak te pakken te krijgen van het programmeren in MSX-basic worden hieronder enkele eenvoudige sleutelwoorden behandeld. Aan het einde van deze paragraaf komen we tot de conclusie dat we het eerste echte computerprogramma in MSX-basic hebben geschreven.

Tik met behulp van de ondertussen vertrouwd geworden MSX-editor eens de volgende regel in:

```
PRINT "HALLO ALLEMAAL"
```

Nadat de RETURN-toets is ingedrukt, zal de computer netjes de tekst "HALLO ALLEMAAL" afdrukken om vervolgens met de Ok-melding aan te geven dat hij klaar is.

We droegen de computer (PRINT betekent AFDRUKKEN) op om een tekst af te drukken; de computer gehoorzaamde ons feilloos.

Probeer ook de volgende regel eens:

```
PRINS "HALLO ALLEMAAL"
```

Iedere BASIC-programmeur begrijpt wat er bedoeld wordt. Echter, de computer, tenslotte maar een dom instrument, begrijpt niet dat het sleutelwoord eigenlijk PRINT had moeten zijn en geeft een syntax error (=fout in schrijfwijze).

Hieruit zal één ding duidelijk zijn:

DE COMPUTER IS EEN DOM INSTRUMENT EN ALS ZODANIG NIET IN STAAT OM FOUTEN TE MAKEN. DE COMPUTER GEHOORZAAMT U FEILLOOS WANNEER U IN STAAT BENT OM DE OPDRACHTEN PRECIES VOLGENS DE VOORSCHRIFTEN IN ZIJN TAAL TE FORMULEREN.

We zien dat we met de regel PRINT "HALLO ALLEMAAL" een eerder gestelde regel in verband met de editor hebben overtreden; we hebben voor deze regel namelijk geen regelnummer geplaatst. Echter, geen foutmelding volgde; de computer begreep onze bedoelingen en voerde het commando feilloos uit.

Wanneer we de regel PRINT "HALLO ALLEMAAL" wèl voorzien van een regelnummer, kunnen we deze regel later met het reeds bekende LIST-commando weer zichtbaar maken. Echter, na het intikken van deze regel voert de computer het commando niet meer uit.

Wanneer we het hele kleine programma dat we nu in feite hebben geschreven, willen laten uitvoeren door de computer, dan moeten we het commando RUN gebruiken. Na dit commando zal de tekst "HALLO ALLEMAAL" weer keurig worden afgedrukt. We kunnen dit enige malen herhalen door steeds weer het RUN-commando te gebruiken. Merk op dat het veel gebruikte RUN-commando standaard onder de vijfde functietoets aanwezig is en het LIST-commando onder de vierde toets kan worden gevonden.

We kunnen het nu ontstane programma uitbreiden met nog een regel:

```
10 PRINT "HALLO ALLEMAAL"
20 PRINT "HOE GAAT HET ER MEE"
RUN
HALLO ALLEMAAL
HOE GAAT HET ER MEE
Ok
```

We zien dat na het RUN-commando de twee opgedragen teksten keurig onder elkaar worden afgedrukt. Blijkbaar werkt de computer de ingetoetste programmaregels op volgorde van regelnummer af totdat alle regels 'op' zijn. We kunnen het programma op deze wijze willekeurig groter maken en steeds weer laten uitvoeren. Een leuk grapje kan men uithalen door het programma als volgt uit te breiden:

```
10 PRINT "HALLO ALLEMAAL"
20 PRINT "HOE GAAT HET ER MEE"
30 GOTO 10
RUN
```

Na het commando RUN blijft de opgedragen tekst achter elkaar

worden afgedrukt op het beeldscherm. Het commando GOTO 10 zorgt ervoor, dat wanneer de computer bij regel 30 is aangekomen, hij weer 'teruggaat' naar regel 10 waardoor het programma geen einde kan vinden. Nogmaals blijkt dat de computer een dom instrument is dat de moeilijkste maar ook de domste en meest onzinnige opdrachten zonder protest uitvoert. De enige voorwaarde is, dat de juiste formulering is gebruikt.

Het steeds maar ronddraaiende programma kan door de STOP-toets tijdelijk worden onderbroken; links onder in het beeld verschijnt dan de cursor. Door de STOP-toets een tweede maal aan te raken, gaat het programma weer verder. Definitief kan het programma worden onderbroken door de toetsen CTRL en STOP tegelijkertijd in te drukken. Met een LIST kunnen we dan het programma weer te voorschijn halen.

Vervang nu regel 10 eens door:

```
10        PRINT"HALLO ALLEMAAL"
```
(denk aan de spaties)

en probeer het programma nog eens.
Conclusie: het MSX-basic is niet erg gevoelig voor spaties; het programma werkt nog precies hetzelfde. We hadden dus net zo goed:

```
10PRINT"HALLO ALLEMAAL"
```
(helemaal geen spaties)

kunnen intikken.
Ook wanneer we:

```
10print "HALLO ALLEMAAL"
```
(kleine letters)

intikken, zien we dat het programma nog steeds uitstekend werkt. We kunnen concluderen dat het gebruik van kleine of grote letters bij sleutelwoorden niets uitmaakt. Bij een LIST zal zelfs blijken dat de computer het woordje print zelf in hoofdletters heeft veranderd.

Het zal duidelijk zijn dat het programma wel anders gaat werken wanneer we de tekst tussen de aanhalingstekens in kleine letters gaan zetten.

We kunnen vaststellen dat:

– sleutelwoorden met kleine en grote letters mogen worden

geschreven.
- sleutelwoorden aan elkaar dienen te worden geschreven.
- er tussen aanhalingstekens precies de tekst dient te worden opgenomen die ook moet worden gebruikt.
- er verder naar believen spatietekens mogen worden gebruikt.

In deze paragraaf kwamen we slechts enkele van de ruim 150 mogelijke sleutelwoorden tegen. In de volgende hoofdstukken worden eerst wat broodnodige begrippen behandeld waarna we in de hoofdstukken 9 en 10 worden geconfronteerd met **alle** bestaande sleutelwoorden met hun preciese schrijfwijzen en betekenissen. Alhoewel de principes van het programmeren vrij eenvoudig zijn, zal het door het grote aantal sleutelwoorden toch een behoorlijke oefening vergen voordat u het MSX-basic volledig beheerst.

5 CONSTANTEN

Om het MSX-basic goed te kunnen beheersen, is begrip nodig van enkele algemene termen. In dit hoofdstuk wordt de term CONSTANTE behandeld.

5.1 Wat is een constante

De definitie van een constante is vrij eenvoudig:

EEN CONSTANTE IS EEN WAARDE DIE NIET
AAN VERANDERING ONDERHEVIG IS.

Wanneer we het over waarden hebben, hebben we het niet alleen over numerieke waarden (waarden die een hoeveelheid aangeven) maar ook over alfanumerieke waarden (tekstwaarden, teksten of STRINGS).

In het volgende voorbeeld zijn enkele constanten genoemd:

123	22.5	JANSSEN
0.124	–100000	DIT IS EEN TEKST
BOEKJE	ABCDEFGHIJ	10203.4

Merk op dat we in de computerwereld in plaats van de decimale komma altijd een punt gebruiken.

In het vorige hoofdstuk zagen we reeds, dat we alfanumerieke constanten door de computer kunnen laten afdrukken. In het reeds eerder genoemde programma:

```
10 PRINT "HALLO ALLEMAAL"
20 PRINT "HOE GAAT HET ERMEE"
30 GOTO 10
```

worden op programmaregel 10 en 20 de alfanumerieke constanten HALLO ALLEMAAL en HOE GAAT HET ERMEE afgedrukt op het beeldscherm. Alfanumerieke constanten dienen altijd tussen aanhalingstekens (") te worden vermeld in een MSX-basicprogramma.

Numerieke constanten behoeven *niet* tussen aanhalingstekens te worden

vermeld. Indien numerieke constanten toch tussen aanhalingstekens worden geplaatst, noemen we deze constanten niet meer numeriek maar ALFANUMERIEK. Het volgende programma:

```
NEW
10 PRINT 125
20 PRINT "125"
```

(eerst oude programmaregels verwijderen)

zal, wanneer het door middel van het commando RUN in werking wordt gezet, twee maal het getal 125 op het beeldscherm afdrukken. Toch noemen we de constante op regel 10 een numerieke constante en de constante op regel 20 een alfanumerieke constante.

5.2 Rekenen met constanten

We kunnen de computer gebruiken als rekenmachine door hem bijvoorbeeld constanten bij elkaar te laten optellen. Tik bijvoorbeeld eens in:

```
PRINT 123+200
```

De computer zal na het intoetsen van de RETURN-toets onmiddellijk met het antwoord, 323 reageren. Ook moeilijkere sommen maakt de computer zonder problemen. Bijvoorbeeld:

```
NEW
10 PRINT 12.445+123.543
20 PRINT 12/6.5
30 PRINT 123*456
40 PRINT 1000-1
```

(eerst de oude programmaregels verwijderen).

Dit voorbeeldprogramma zal na het RUN-commando in een oogwenk de uitkomsten van de opgegeven sommen op het beeldscherm laten zien. Merk op dat het sterretje(*) als vermenigvuldigingsteken en de schuine streep(/) als deelteken worden gebruikt.

Het rekenen gaat natuurlijk alleen met NUMERIEKE constanten. Wanneer we

```
PRINT "JAN"-"PIET"
```

intoetsen, dan zal de computer antwoorden met een foutmelding; de

berekening is niet uit te voeren.

Indien we echter de regel:

```
PRINT "HALLO ALLEMAAL"+"HOE GAAT HET
ER MEE"
```

intoetsen, dan zal de computer de uitkomst HALLO ALLEMAAL HOE GAAT HET ERMEE afdrukken. Blijkbaar kan de computer twee alfanumerieke constanten wel *optellen*. Conclusie:

DE COMPUTER KAN REKENEN MET NUMERIEKE CONSTANTEN. ALFANUMERIEKE CONSTANTEN KUNNEN SLECHTS WORDEN OPGETELD; HET RESULTAAT IS DAN DE AANEENSCHAKELING VAN DEZE CONSTANTEN.

Aan de ingewikkeldheid van een opgedragen som zijn hoegenaamd geen grenzen. De opdracht:

```
PRINT 12*(2/3.5)-11*(12345+1.22)/13
```

zal door de computer in een fractie van een seconde worden uitgevoerd. Ook de opdracht:

```
PRINT "ABCDEFGH"+"IJKLMNOPQ"+"RSTUVWXYZ"
```

wordt onmiddellijk uitgevoerd, waarbij de uitkomst wordt gevormd door het alfabet.

5.3 Soorten van constanten

Het MSX-basic kent diverse soorten van constanten. Sommige soorten constanten bieden geen enkele moeilijkheid, andere zijn zeer lastig te begrijpen.

Een eerste onderverdeling van constanten zagen we reeds: de onderverdeling tussen NUMERIEKE en ALFANUMERIEKE constanten.

Van alfanumerieke constanten, die we vaak STRING-CONSTANTEN (string betekent keten) noemen, is er maar één soort. Van numerieke constanten zijn er echter meerdere soorten die we per stuk in de volgende paragrafen zullen behandelen.

5.4 Integere constanten

De integere constante is een eenvoudige vorm van numerieke constanten. Een integere constante dient te voldoen aan de volgende eisen:

- de integere constante mag niet kleiner zijn dan –32768
- de integere constante mag niet groter zijn dan 32767
- de integere constante mag geen decimalen en ook geen decimale punt bevatten

5.5 Constanten met enkelvoudige precisie

De constante met enkelvoudige precisie heeft een veel grotere vrijheid in het gebruik. De volgende eisen worden aan een constante met enkelvoudige precisie gesteld:

- de minimum waarde voor een constante met enkelvoudige precisie is $-9.99... \times 10^{62}$
- de maximum waarde voor een constante met enkelvoudige precisie is $9.99... \times 10^{62}$
- de grootst mogelijke waarde kleiner dan 0 voor een constante met enkelvoudige precisie is -10^{-64}
- de kleinst mogelijke waarde groter dan 0 voor een constante met enkelvoudige precisie is 10^{-64}
- een constante met enkelvoudige precisie heeft een precisie (significantie) van 6 cijfers

Omdat de exponentiële notatie vooral bij grotere of kleinere getallen het voordeel biedt dat het een korte notatie is (probeer het getal 1.2×10^{60} maar eens gewoon uit te schrijven...) heeft Microsoft deze notatie ook in het MSX-basic mogelijk gemaakt voor deze constanten. De notatievorm moest echter natuurlijk enigszins aan de mogelijkheden van de computer worden aangepast.

De constante 1.0×10^3 wordt in MSX-basic genoteerd als 1.0E3 of 1.0D3. Zo wordt 1.243×10^{16} bijvoorbeeld genoteerd als 1.243E16 of 1.1234D16 en wordt 3.14×10^{-33} genoteerd als 3.14E–33 of 3.14D–33.

Achter de mantisse wordt in de MSX-notatie (zoals in veel andere BASIC-talen) eerst een D of een E geplaatst waarna de macht van de exponent wordt opgenomen. Het grondtal van de exponent is bij deze notatie altijd gelijk aan 10 en wordt weggelaten.

5.6 Constanten met dubbele precisie

Het enige verschil tussen constanten met dubbele precisie en constanten met enkelvoudige precisie is, dat een constante met dubbele precisie geen precisie van 6 cijfers maar een precisie van 14 cijfers heeft.

Om aan te geven dat een constante een dubbele precisie heeft,
- moet achter de constante een hekje (#) worden geplaatst, of:
- moet het exponentgedeelte, indien aanwezig, de exponentaanduiding D bevatten. De exponentaanduiding E is voor deze constante *verboden*.

5.7 Bij twijfel...

Het is niet verplicht om bij een constante een achtervoegsel (#,% of !) te gebruiken. Bij sommige constanten is het dan echter niet precies uit te maken of het nu een integere constante, een constante met enkelvoudige precisie of een constante met dubbele precisie is. Bij de exponentiële notatie bestaat de twijfel nooit; uit de in de exponent genoteerde letter (D of E) is zonder meer het type constante af te leiden.

Indien het MSX-basic het verschil niet precies kan bepalen, dan maakt het om misverstanden te voorkomen, altijd een constante met dubbele precisie van de constante waarover wordt getwijfeld. Dit is een veilige beslissing van het MSX-basic; berekeningen kunnen hierdoor moeilijk verkeerd uitpakken.

5.8 Binaire constanten

Een binaire constante dient aan de volgende voorwaarden te voldoen:
- de constante dient door &B te worden voorafgegaan.
- de constante mag alleen de cijfers 0 en 1 bevatten.
- de constante mag maximaal gelijk aan &B1111111111111111 zijn (65535 decimaal).

5.9 Octale constanten

Een octale constante dient aan de volgende voorwaarden te voldoen:

- de constante dient door &O te worden voorafgegaan.
- de constante mag alleen de cijfertekens 0...7 bevatten.
- de constante mag maximaal gelijk zijn aan &O177777(65535 decimaal).

5.10 Hexadecimale constanten

Een hexadecimale constante dient aan de volgende eisen te voldoen:

- de constante dient door &H te worden voorafgegaan.
- de constante mag alleen de cijfertekens 0...9 en A...F bevatten.
- de constante mag maximaal gelijk zijn aan &HFFFF (65535 decimaal).

6.1 Wat is een variabele

Ook de definitie van een variabele is vrij eenvoudig:

EEN VARIABELE IS EEN WAARDE DIE (IN TEGENSTELLING TOT EEN CONSTANTE) WEL AAN VERANDERINGEN ONDERHEVIG IS.

Een variabele kan dus niet met één vaste aanduiding worden genoteerd. In plaats daarvan dienen we variabelen een naam te geven. Wanneer we bijvoorbeeld intikken:

```
WAARDE=12.5
```

Dan hebben we op dat moment aan de variabele WAARDE de constante 12.5 toegekend. Wanneer we later dan intoetsen:

```
PRINT WAARDE
```

dan zal de computer vervolgens de aan de variabele WAARDE toegekende waarde weer projecteren.

Wanneer we vervolgens intikken:

```
HOEVEELHEID=26
```

dan is op dat moment de waarde 26 aan variabele HOEVEELHEID toegekend. Tikken we nu in:

```
PRINT WAARDE+HOEVEELHEID
```

Dan zal de computer netjes de twee eerder aan de variabelen toegekende WAARDEN en HOEVEELHEID bij elkaar optellen en het resultaat op het beeldscherm laten zien.

We kunnen de waarde van een variabele ook herzien. Indien we intikken:

```
HOEVEELHEID=622.5
```

En daarna:

```
PRINT HOEVEELHEID+WAARDE
```

dan zien we, dat de variabele HOEVEELHEID een andere waarde heeft verkregen. De variabele HOEVEELHEID maar natuurlijk ook alle andere variabelen zijn dus onderhevig aan veranderingen.

Bovenstaande voorbeelden behandelen slechts één van de typen variabelen, namelijk de numerieke variabelen. We kennen echter ook een heel ander soort variabelen, namelijk de alfanumerieke variabelen.

Laten we bijvoorbeeld eens intikken:

```
LET NAAM$="GEERT-JAN"
```

Het woordje LET betekent: laat of laat zijn. LET NAAM$= "GEERT-JAN" betekent dus zoveel als laat variabele NAAM$ gelijk zijn aan GEERT-JAN. Het sleutelwoord LET zouden we ook in onze eerdere voorbeelden kunnen hebben gebruikt maar het mag in deze constructie zonder meer worden weggelaten. We hadden dus ook gewoon:

```
NAAM$="GEERT-JAN"
```

kunnen intikken.

De variabele NAAM$ is een alfanumerieke variabele. Deze heeft dan ook verplicht een dollarteken ($) als laatste teken. Alfanumerieke variabelen kunnen allerlei karakters bevatten. Net als bij alfanumerieke constanten kunnen we alfanumerieke variabelen alleen maar optellen. Probeer bijvoorbeeld eens in te tikken:

```
PRINT NAAM$
```

en

```
PRINT NAAM$+" IS MIJN NAAM."
```

In het laatste voorbeeld werden een alfanumerieke variabele en een alfanumerieke constante bij elkaar opgeteld. Het resultaat bestond uit een aaneenschakeling van deze variabele en constante.

Om het begrip variabele nog wat verder uit te diepen, hebben we de

medewerking van nog een MSX-sleutelwoord nodig, namelijk het sleutelwoord INPUT. Met dit sleutelwoord is het mogelijk om een variabele een waarde te geven door intoetsing terwijl het programma werkt. Tik bijvoorbeeld het volgende programma eens in:

```
NEW    (om oude programmaregels te verwijderen)
10 PRINT "REKENPROGRAMMA"
20 INPUT "HOE HEET U ";NAAM$
30 PRINT "GOEDENDAG "+NAAM$+"HOE IS HET"
40 INPUT "GEEF EEN GETAL IN ";GETAL
50 PRINT GETAL+GETAL;" IS HET DUBBELE
VAN ";GETAL
60 GOTO 40
```

Zo hebben we met relatief weinig sleutelwoorden al een heel programma geschreven. In regel 10 wordt op het beeldscherm vermeld dat het hier om een rekenprogramma gaat. Niets bijzonders, slechts een eenvoudige alfanumerieke constante wordt op het beeldscherm afgedrukt.

In regel 20 komt het nieuwe sleutelwoord, INPUT, aan de beurt.

Wanneer we het programma met een RUN in werking stellen, zien we dat de tekst HOE HEET U netjes op het beeldscherm wordt afgedrukt en dat de computer daarna wacht. De computer verwacht nu namelijk van u, dat u een alfanumerieke constante intoetst en dat u, zoals altijd, deze ingave met een RETURN-toets afsluit. Uw ingave wordt in variabele NAAM$ bewaard.

In regel 30 presenteert de computer de optelling van een alfanumerieke constante, een alfanumerieke variabele en weer een alfanumerieke constante. Hierdoor begroet de computer ons hoffelijk en noemt hij ons zelfs bij de naam; een bewijs dat in NAAM$ inderdaad de eerder gepleegde ingave is geplaatst.

In regel 40 vraagt de computer weer om een ingave. Dit maal dient de ingave een numerieke constante te zijn; dat is te zien aan de naam van de variabele in het INPUT-commando. Deze naam eindigt niet op een dollarteken en dit duidt op een numerieke variabele. Indien we tijdens deze INPUT proberen, een niet geldige numerieke constante in te geven, dan geeft de computer een foutmelding en krijgen we nogmaals de kans om een correcte numerieke constante in te geven.

In regel 50 wordt de optelling van de variabele GETAL bij zichzelf afgedrukt. Op het beeldscherm verschijnt het dubbele van de ingegeven

waarde. Na deze waarde wordt de alfanumerieke constante IS HET DUBBELE VAN afgedrukt, gevolgd door de waarde van de inhoud van numerieke variabele GETAL. Uit deze programmaregel leren we meteen dat we in een PRINT-commando verschillende zaken achter elkaar aan kunnen afdrukken door deze met een punt-komma van elkaar te scheiden.
Tenslotte gaven we de computer op regel 60 het commando om weer naar programmaregel 40 terug te gaan; de computer vraagt weer om een ingave. Wanneer we de computer niet met CTRL-STOP onderbreken, zal hij voortdurend met dit programma bezig blijven.

We zien dat het gebruik van variabelen een nieuwe dimensie geeft aan de computer. Door het gebruik van variabelen kunnen we eenzelfde programma steeds van andere waarden voorzien waardoor steeds nieuwe situaties worden berekend.

6.2 Soorten van variabelen

Net als bij constanten onderscheiden we ook bij variabelen enkele verschillende typen. De eerste onderverdeling, die tussen numerieke en alfanumerieke variabelen, zagen we reeds. Van de stringvariabelen of alfanumerieke variabelen is er slechts één type. Echter, van de numerieke variabelen onderscheiden we verschillende typen die in de volgende paragrafen worden behandeld.

6.3 De integere variabele

Een integere variabele herkennen we aan het procentteken (%) dat achter de variabelenaam is vermeld. Een integere variabele kan alleen een integere waarde bevatten (zie de beschrijving van integere constanten). Wanneer we proberen om een andere dan integere waarde aan een integere variabele toe te kennen, dan wordt de waarde aangepast of volgt een foutmelding.

6.4 De variabele met enkelvoudige precisie

Een variabele met enkelvoudige precisie kunnen we herkennen aan het uitroepteken (!) dat achter de variabelenaam is vermeld. Een variabele met enkelvoudige precisie is aan dezelfde beperkingen gebonden als een constante met enkelvoudige precisie tot zover van toepassing.

Wanneer we proberen om een waarde, afwijkend van deze bepalingen, aan een variabele met enkelvoudige precisie toe te kennen dan wordt de waarde aangepast of krijgen we een foutmelding.

6.5 De variabele met dubbele precisie

Een variabele met dubbele precisie kunnen we herkennen aan het hekje (#) dat achter de variabelenaam is geplaatst. Een variabele met dubbele precisie is aan dezelfde beperkingen gebonden als de constante met dubbele precisie tot zover van toepassing. Wanneer we proberen om een waarde, afwijkend van deze bepalingen, aan een variabele met dubbele precisie toe te kennen dan wordt de waarde aangepast of krijgen we een foutmelding.

6.6 Bij twijfel...

Indien een variabelenaam geen dollarteken, hekje, uitroepteken of procentteken als laatste karakter heeft, dan is het niet meer duidelijk om wat voor soort variabele het hier gaat. De volgende regels gelden dan:

- in het normale geval neemt de computer voor zo'n waarde aan, dat het om een variabele met dubbele precisie gaat.
- indien echter door middel van een DEFINT, DEFSTR, DEFSNG of DEFDBL sleutelwoord een ander type is verbonden aan de eerste letter van de variabelenaam, dan wordt dat type automatisch aangenomen.

6.7 Array-variabelen

Het kan voorkomen dat we meerdere waarden onder één en dezelfde variabelenaam wilen bewaren. In dat geval zullen we behalve de variabelenaam ook dienen aan te geven welke waarde van alle waarden die we hieronder hebben opgeslagen we bedoelen. Een variabele waaronder we meer dan één waarde willen bewaren, kunnen we vergelijken met een tabel. We noemen zo'n variabele een array-variabele of kortweg een array (rij).
Zo'n array dienen we een bepaalde grootte (een bepaald aantal mogelijkheden tot opslag van waarden) toe te kennen. Dit doen we met behulp van het sleutelwoord DIM. We gaan een programma schrijven:

```
NEW
10 DIM TABEL(3)
```

Wanneer we dit programma in werking stellen, dan wordt een variabele, genaamd TABEL, toegewezen. Deze variabele biedt mogelijkheid tot opslag van *vier* waarden, in dit geval numerieke waarden met dubbele precisie. Deze *vier* waarden worden opgeslagen onder:

TABEL(0)	...eerste waarde
TABEL(1)	...tweede waarde
TABEL(2)	...derde waarde
TABEL(3)	...vierde waarde

In het verdere programma kunnen we deze tabel gaan vullen:

```
20 LET TABEL(0)=12
30 LET TABEL(1)=96
40 LET TABEL(3)=-1E22
50 LET TABEL(2)=TABEL(3)*TABEL(1)/TABEL
(0)
60 PRINT TABEL(0),TABEL(1),TABEL(2),TAB
EL(3)
```

Wanneer we dit programma laten werken via een RUN, dan merken we na enige narekening dat de variabele TABEL inderdaad vier afzonderlijke waarden bevat die we kunnen veranderen en waarmee we kunnen rekenen.

In programmaregel 60 zien we overigens, dat bij een PRINT-commando de afzonderlijke gegevens met een komma van elkaar mogen worden gescheiden. We zagen zoiets al in paragraaf 6.1, maar daar gebruikten we een punt-komma. Het gebruik van een komma heeft tot gevolg dat de gegevens ietwat uit elkaar, zo mogelijk op dezelfde regel worden afgedrukt.

7.1 Wat is een uitdrukking

We kunnen in MSX-basic de computer bijvoorbeeld de volgende som laten oplossen:

```
PRINT 2*(WAARDE+5.5)/(2+AANTAL)
```

De variabelen WAARDE en AANTAL dienen dan wel te zijn gevuld met bepaalde waarden.

De opgave (de som) die na het PRINT-commando staat, noemt men in de computerwereld een UITDRUKKING. Vormen van uitdrukkingen zijn bijvoorbeeld ook:

2+3	A/B	"JAN"+"KAREL"
25/(3–4*Q)	"HALLO"+NAAM$	12.55+3%

Deze uitdrukkingen, numeriek of alfanumeriek, geven aan:

- welke bewerkingen dienen te geschieden (optellen, aftrekken, vermenigvuldigen etcetera)
- op welke variabelen deze bewerkingen dienen te worden toegepast
- op welke constanten deze bewerkingen dienen te worden toegepast
- in welke volgorde de bewerkingen dienen te worden toegepast.

In feite kunnen we een uitdrukking beschouwen als een som die wij de computer opgeven. Zo'n uitdrukking noemen we numeriek indien het resultaat van deze uitdrukking numeriek is. We noemen een uitdrukking alfanumeriek indien het resultaat van deze uitdrukking alfanumeriek is. In een uitdrukking vinden we de volgende elementen:

- variabelen. Deze hebben een waarde die al eerder bepaald is.
- constanten. Deze hebben een vastgestelde waarde.
- bewerkingscodes. Deze geven aan welke bewerkingen dienen te geschieden.
- voorrangstekens. Deze worden altijd gevormd door haakjes. Met deze voorrangstekens kan een afwijkende voorrang in bewerking worden vastgelegd.

We kunnen een uitdrukking als volgt definiëren:

EEN UITDRUKKING IS EEN IN PRINCIPE UITVOERBAAR SAMENSTELSEL VAN CONSTANTEN, VARIABELEN, BEWERKINGEN EN VOORRANGSTEKENS.

Op de eerste twee elementen, de constanten en variabelen, zijn we in de voorgaande hoofdstukken reeds diep ingegaan. In de volgende paragrafen gaan we op bewerkingen en voorrang en voorrangstekens in.

7.2 Algebraïsche bewerkingen

Het MSX-basic kent de volgende algebraïsche bewerkingen:

- \+ optellen. Met het plus-teken geven we aan dat de uitdrukkingen links en recht van dit teken bij elkaar dienen te worden opgeteld. Dit is de enige bewerking die ook op alfanumerieke waarden mag worden toegepast.
- – aftrekken. Met het min-teken geven we aan dat de uitdrukkingen links en rechts van dit teken van elkaar moeten worden afgetrokken. Deze bewerking mag alleen op numerieke bewerkingen worden toegepast.
- * vermenigvuldigen. Met het sterretje of vermenigvuldigingsteken geven we aan dat de uitdrukkingen rechts en links van dit teken met elkaar dienen te worden vermenigvuldigd. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.
- / delen. Met de deelstreep geven we aan dat de uitdrukkingen links en rechts van dit teken door elkaar moeten worden gedeeld. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.
- ^ machtsverheffen. Met het 'dakje' geeft men aan, dat de uitdrukking links van dit teken in een macht dient te worden verheven. De uitdrukking rechts van dit teken geeft aan om welke macht het gaat. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.
- \ integer delen. Met deze omgedraaide deelstreep geeft men aan dat de uitdrukkingen links en rechts van dit teken eerst dienen te worden gemaakt tot een integere waarde (een gehele waarde, minimaal gelijk aan –32768 en maximaal gelijk aan 32767). Vervolgens worden deze integere waarden op elkaar gedeeld. De uitkomst wordt een tweede maal gemaakt tot

een integere waarde. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.

MOD restbepaling. Met dit sleutelwoord geeft men aan dat de uitdrukkingen links en rechts van dit sleutelwoord eerst dienen te worden gemaakt tot een integere waarde. Daarna dienen deze twee integere waarden door elkaar te worden gedeeld. De restwaarde die bij deling overblijft (er wordt in dit geval niet na de decimale punt doorgedeeld) is de uitkomst van deze bewerking. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.

7.3 Functionele bewerkingen

Een functionele bewerking is een bewerking die kan worden toegepast op een uitdrukking of op een stelsel van uitdrukkingen. Een functionele bewerking heeft altijd de vorm van:

FUNCTIONELE BEWERKING (UITDRUKKING...)

Om het één en ander wat nader toe te lichten, volgen hieronder enkele voorbeelden van functionele bewerkingen.

Tik bijvoorbeeld eens in:

```
PRINT INT(12.9)
```

De computer zal antwoorden met het getal 12. De functionele bewerking INT heeft tot gevolg dat het grootste gehele getal kleiner dan de tussen de haakjes staande uitdrukking wordt berekend.

```
PRINT LEFT$("ABCDEFGHIJKLMNOP",5)
```

De computer antwoordt hier met de uitkomst ABCDE. De functie LEFT$ heeft tot gevolg dat het linkergedeelte van de opgegeven alfanumerieke uitdrukking wordt bepaald. Het aantal tekens is gegeven in de tweede tussen haakjes gegeven uitdrukking, die numeriek is.

```
PRINT SGN(-12.8)
```

De computer antwoordt hier met de uitkomst –1. De functie SGN geeft drie waarden, namelijk 1 (indien de uitdrukking tussen haakjes positief is), 0 (indien de uitdrukking tussen haakjes gelijk is aan 0) of

–1 (indien de uitdrukking tussen haakjes negatief is).

7.4 Relationele bewerkingen

Het MSX-basic kent een aantal relationele bewerkingen. Een relationele bewerking is een bewerking die uitdrukking geeft aan een relatie.

Tik bijvoorbeeld eens in:

```
PRINT 2=2
```

De computer zal het antwoord –1 geven. Dit antwoord betekent dat de bewering 2=2 inderdaad waar is. De uitdrukking links van het gelijkteken en de uitdrukking rechts van het gelijkteken hebben de relatie, dat ze in waarde gelijk zijn aan elkaar. Tik bijvoorbeeld eens in:

```
PRINT (5+2)=(8-1)
```

Ook hier zal het oordeel van de computer WAAR (–1) luiden; 5+2 is inderdaad gelijk aan 8–1. Tik nu eens in:

```
PRINT 5+3=7
```

Het oordeel van de computer is nu 0 (NIET WAAR). De uitdrukkingen 5+3 en 7 hebben niet de relatie gelijkheid met elkaar.

Het gelijkteken in de hiervoor gegeven voorbeelden is een relationele bewerking. Een relationele bewerking dient altijd plaats te vinden tussen twee numerieke uitdrukkingen of tussen twee alfanumerieke uitdrukkingen. Het resultaat is altijd –1 (WAAR) of 0 (NIET WAAR).

De bewerking 'is gelijk aan' (=) is slechts één van de relationele bewerkingen die het MSX-basic kent. Hieronder volgt een tabel met alle relationele bewerkingen:

RELATIONELE BEWERKING	BETEKENIS
=	is gelijk aan
<	is kleiner dan
>	is groter dan
<> OF ><	is kleiner of groter dan (ofwel: is ongelijk aan)
<= OF =<	is kleiner dan of gelijk aan (ofwel: is ten hoogste gelijk aan)
>= OF =>	is groter dan of gelijk aan (ofwel: is ten minste gelijk aan)

7.5 Logische bewerkingen

Het MSX-basic biedt verscheidene logische bewerkingen. Logische bewerkingen dienen ondermeer om zeer ingewikkelde afvragingen te verrichten.
De beginnende programmeur kan, zeker in eerste instantie, misschien deze paragraaf beter overslaan.

Een logische bewerking vergelijkt twee uitdrukkingen en doet vervolgens een uitspraak in de zin van WAAR (ongelijk aan 0) of NIET WAAR (gelijk aan 0). Aan de hand van de meest gemakkelijke logische bewerking lichten we dit wat nader toe. Tik bijvoorbeeld eens in:

eerste regel:

```
IF 2+3=5 AND 4+6=10 THEN PRINT "HOERA"
```

tweede regel:

```
IF 2+4=6 AND 6+6=11 THEN PRINT "EEN
VREEMDE ZAAK"
```

derde regel:

```
IF "JAN"<"KAREL" AND 4*4=16 THEN PRINT
"HOERA"
```

De eerste regel resulteert evenals de derde regel in het afdrukken van de tekst HOERA op het beeldscherm. De tweede regel blijft zonder resultaat. Verklaring: De logische bewerking AND onderzoekt de uitdrukking links en rechts van deze bewerking. Alleen indien beide bewerkingen WAAR zijn (–1 als uitkomst hebben) dan krijgt de totale uitdrukking tussen IF en THEN de beoordeling WAAR (de waarde –1) en wordt het gedeelte achter THEN uitgevoerd. De derde regel geeft een extra moeilijkheid; er wordt een bewering gedaan met twee teksten waarbij wordt beweerd dat de eerste tekst kleiner is dan de tweede tekst. Indien de uitdrukkingen waarop een relationele bewerking wordt toegepast, alfanumeriek zijn, dan worden deze uitdrukkingen alfabetisch uitgewaardeerd. Een tekst die bij een alfabetische rangschikking voor een tweede tekst terecht zou komen, is ook kleiner dan deze tweede tekst.

Over het algemeen kan men stellen dat karakters volgens de MSX-karaktertabel (hoofdstuk 18) worden vergeleken waarbij een karakter met de laagst gecodeerde waarde ook het kleinst is.

De eerste voorbeeldregel zou in normaal Nederlands als volgt kunnen worden vertaald:

Als 2+3 gelijk is aan 5 *en* 4+6 is gelijk aan 10, druk dan de tekst HOERA af.

De logische bewerking AND kan men dus vertalen met het woordje EN.

Het MSX-basic kent diverse logische bewerkingen. Hieronder volgt een tabel met alle logische bewerkingen die in MSX-basic bestaan:

logische bewerking	betekenis ongeveer	uitleg
AND	en	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft de uitkomst WAAR (ongelijk aan nul) alleen indien de beide uitdrukkingen WAAR (ongelijk aan nul) zijn.
OR	of	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft de uitkomst WAAR indien minstens één van de twee uitdrukkingen WAAR is.
XOR	exclusief of	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft slechts de uitkomst WAAR indien minstens en hoogstens één van de uitdrukkingen WAAR is.
EQV	gelijkwaardig met	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft slechts de uitkomst WAAR indien beide uitdrukkingen WAAR zijn of beide uitdrukkingen NIET WAAR zijn (wanneer beide uitdrukkingen dus gelijkwaardig zijn).
IMP	impliceert dat	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft slechts de uitkomst NIET WAAR indien de eerste uitdrukking waar is maar de tweede niet (wanneer het WAAR zijn van de eerste uitdrukking niet impliceert dat de tweede uitdrukking WAAR is).
NOT	niet	deze logische bewerking wordt vóór een tussen haakjes gestelde uitdrukking geplaatst en draait de waarde van deze uitdrukking om. Indien deze uitdrukking WAAR is, veroorzaakt deze bewerking dus het antwoord NIET WAAR en omgedraaid.

7.6 Waarheidstabel

De volgende waarheidstabel geldt voor de logische bewerkingen (0 = niet waar en 1 = waar):

bewerking	eerste argument	tweede argument	resultaat
NOT	0		1
	1		0
AND	0	0	0
	0	1	0
	1	0	0
	1	1	1
OR	0	0	0
	0	1	1
	1	0	1
	1	1	1
XOR	0	0	0
	0	1	1
	1	0	1
	1	1	0
EQV	0	0	1
	0	1	0
	1	0	0
	1	1	1
IMP	0	0	1
	0	1	1
	1	0	0
	1	1	1

7.7 Voorrangsregels voor bewerkingen

Haakjes hebben in een bewerking altijd de allerhoogste voorrang. Buiten deze haakjes liggen de voorrangsregels als volgt:

voorrangs-volgorde	bewerkingen	opmerkingen	
1	functionele bewerkingen	Door het verplichte gebruik van haakjes kunnen onderling geen voorrangsconflicten ontstaan.	
2	algebraïsche bewerkingen	eerst ^	
		dan *, /, \ en MOD	1)
		dan + en –	1)
3	relationele bewerkingen		1)
4	logische bewerkingen		1)

1) Indien er tussen gelijkwaardige bewerkingen conflictsituaties ontstaan, dan worden deze bewerkingen van links naar rechts in volgorde uitgevoerd.

8 DE SLEUTELWOORDEN VAN MSX-BASIC

In dit hoofdstuk worden de belangrijkste sleutelwoorden van het MSX-basic behandeld.

De sleutelwoorden worden in alfabetische volgorde behandeld.
Per sleutelwoord worden de volgende gegevens verstrekt:

- de naam van het sleutelwoord

- de soort. We onderscheiden binnen de sleutelwoorden de A-FUNKTIES (functies met alfanumeriek resultaat), de N-FUNKTIES (functies met numeriek resultaat), de SYSTEEMVARIABELEN (door MSX vastgestelde variabelen) en de KOMMANDO'S (sleutelwoorden die een aktie aangeven)

- de schrijfwijze. De toegestane syntaxis of schrijfwijze is in een verkorte BNF-notatie opgenomen voor elk sleutelwoord. De volgende afkortingen zijn hierbij gebruikt:

  ⟨A⟩... alfanumerieke uitdrukking
  ⟨N⟩... numerieke uitdrukking
  ⟨U⟩... uitdrukking

- de betekenis. De functie van elk sleutelwoord wordt behandeld

- voorbeelden. Waar zinvol zijn voorbeelden opgenomen. De voorbeelden zijn waar nodig in verband met het gebruik van andere sleutelwoorden *gebaseerd op de aanbevolen leervolgorde.*

SLEUTELWOORD ABS

soort . . . . **N-FUNKTIE**

schrijfwijze

ABS(<N>)

betekenis

Deze funktie geeft van de tussen haakjes vermelde numerieke uitdrukking de ABSOLUTE WAARDE. De absolute waarde van een getal is de waarde van dit getal, ontdaan van enig teken.

Zo is bijvoorbeeld de absolute waarde van:

het getal 12.347	het getal 12.347
en van het getal -133.5	het getal 133.5

Voorbeeld:

```
PRINT ABS(7*(-5))
 35
Ok
PRINT ABS(111)
 111
Ok
```

SLEUTELWOORD ASC

soort . . . . **N-FUNKTIE**

schrijfwijze

ASC(<A>)

betekenis

Deze funktie geeft de ASCII-waarde van het eerste teken van de tussen haakjes vermelde alfanumerieke uitdrukking als resultaat.

De ASCII-waarde is de waarde die de computer intern in het geheugen gebruikt om allerlei tekens op te slaan. Deze waarde is nooit kleiner dan 0 en nooit groter dan 255.

Zie hoofdstuk 17 en hoofdstuk 18. In hoofdstuk 17 staat de standaard ASCII-tabel, in hoofdstuk 18 staat de daarvan afgeleide MSX-tabel.

Voorbeeld:

```
NEW
10 LET G$="MSX"
20 PRINT ASC(G$)
RUN
 77
Ok
```

(77 is de ASCII-kodering voor de letter M)

SLEUTELWOORD ATN

---

soort . . . . **N-FUNKTIE**

schrijfwijze

<u>ATN</u>(<N>)

betekenis

Deze funktie geeft de arctangens van de tussen haakjes vermelde numerieke uitdrukking als resultaat. Het resultaat ligt altijd tussen $-\frac{1}{2}\pi$ en $\frac{1}{2}\pi$ en drukt als zodanig een hoek in radialen uit.

Voorbeeld:

```
NEW
10 INPUT "WAARDE ";A
20 PRINT "DE ARCTANGENS VAN";A;"IS";AT
N(A)
RUN
WAARDE ? 22
DE ARCTANGENS VAN 22 IS 1.5253730473733
Ok
```

SLEUTELWOORD # AUTO

---

soort .... **KOMMANDO**

schrijfwijze

```
AUTO[<REGELNUMMER>][,[<STAPGROOTTE>]]
<REGELNUMMER>::=<TYPE 1 INTEGERE KON-
  STANTE>
<STAPGROOTTE>::=<TYPE 1 INTEGERE KON-
  STANTE>
<TYPE 1 INTEGERE KONSTANTE>::=<ZIE AL-
  GEMENE SPECIFICATIES>
```

betekenis

Dit kommando kan bij invoer van lange programma's erg nuttig zijn. Het verstrekt namelijk automatisch regelnummers; alleen de kommando's behoeven nog maar te worden ingetoetst. Voorbeeld:

```
NEW
AUTO
10 PRINT "AUTO-TEST"
20
```

– Etcetera, de regelnummers worden automatisch gegeven, alleen de kommando's behoeven te worden ingetoetst.

AUTO kan worden onderbroken door CONTROL-STOP of CONTROL-C.

Indien een regel dient te worden overgeslagen, behoeft alleen een RETURN te worden gegeven; een programmaregel met het overgeslagen nummer wordt dan niet in het programma opgenomen.

Indien een programmaregel reeds bestaat, geeft AUTO dat aan door middel van een ster-teken (*). Bijvoorbeeld (uitgaande van het eerste stukje programma):

```
AUTO
10*PRINT "TWEEDE TEST"
20 REM NOG EEN REGEL
30
```

(Regel 10 was reeds aanwezig en is nu overgetikt.)

Indien regelnummers met sterren worden overgeslagen door alleen RETURN in te toetsen, blijft de oude programmaregel gehandhaafd.

AUTO kan worden vergezeld van een regelnummer en een stapgrootte. Bijvoorbeeld:

```
AUTO
AUTO 100
AUTO 5,2
AUTO ,2
```

Vanaf regel 10 worden regelnummers met stappen van 10 gegeven.
Vanaf regel 100 worden regelnummers met stappen van 10 gegeven.
De regels 5,7,9,11... worden gegeven.
De regels 0,2,4,... worden gegeven.

Het AUTO-kommando mag, alhoewel dat niet erg zinvol is, in een programmaregel worden opgenomen. RENUM (zie de behandeling van dit kommando) beschouwt ten onrechte echter de achter AUTO opgenomen waarden als regelnummers en probeert deze bij het hernummeren ook mee te nemen, hetgeen vreemde resultaten kan opleveren.

Merk op dat AUTO standaard onder funktietoets 2 aanwezig is.

SLEUTELWOORD # BEEP

soort . . . . **KOMMANDO**

schrijfwijze

BEEP

betekenis

Dit kommando veroorzaakt een kort piep-signaal, te gebruiken bij bijvoorbeeld een INPUT als attentiesignaal. Voorbeeld:

```
NEW
10 BEEP:INPUT "UW NAAM ";A$
20 PRINT "GOEDENDAG, ";A$
30 STOP
RUN
UW NAAM ? FREDERIK              (een attentiesignaal klinkt)
GOEDENDAG, FREDERIK
Break in 30
Ok
```

Indien u een televisieaansluiting heeft, komt het geluid via uw televisieluidspreker. U dient het volume van uw toestel dus wel wat open te draaien.

SLEUTELWOORD **BIN$**

soort . . . . **A-FUNKTIE**

schrijfwijze

BIN$(<N>)

betekenis

Deze funktie geeft de binaire waarde van de tussen haakjes vermelde uitdrukking weer in string-vorm. De waarde van de numerieke uitdrukking dient te liggen tussen – 32769 en 65536. Indien de waarde van de numerieke uitdrukking gebroken is, wordt een decimale fractie verwaarloosd.
Indien de waarde van de numerieke uitdrukking negatief is, geeft BIN$ een binaire representant, samengesteld volgens de tweecomplementmethode. Bijvoorbeeld:

```
PRINT BIN$(-32768)
1000000000000000
Ok
PRINT BIN$(65535)
1111111111111111
Ok
PRINT BIN$(-1)
1111111111111111
Ok
A$=BIN$(1023)
Ok
PRINT A$
1111111111
Ok
```

SLEUTELWOORD # CHR$

soort . . . . **A-FUNKTIE**

schrijfwijze

```
CHR$(<ASCII-KODE>)
<ASCII-KODE>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Deze funktie geeft als resultaat de letter die korrespondeert met de gegeven ASCII-kode. De ASCII-kode is de interne kode die de computer gebruikt om allerlei tekens in het geheugen op te slaan en moet minimaal gelijk zijn aan nul en maximaal gelijk zijn aan 255. Een eventuele fractie wordt verwaarloosd.

Zie voor de ASCII-kodering hoofdstuk 17 en voor de hiervan afgeleide MSX-tabel hoofdstuk 18.

Voorbeeld:

```
NEW
10 FOR I=65 TO 90
20 PRINT CHR$(I);
30 NEXT I
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ok
```

(de ASCII-kodes 65 ... 90 corresponderen met de letters A ... Z).

SLEUTELWOORD **CIRCLE**

soort .... **KOMMANDO**

schrijfwijze

```
CIRCLE<LOCATIE>,<STRAAL>[,[<KLEUR>][,
  [<AANVANGSHOEK>][,[<EINDHOEK>][,<AF-
  PLATTING>]]]]\CIRCLE<...>,{,}
  <LOKATIE>::=[STEP](<HORIZONTAAL>,<VER-
  TIKAAL>)
```

betekenis

Met dit kommando kunnen cirkels, ellipsen, cirkeldelen en ellipsdelen worden getekend. Voor goed begrip van dit kommando is het vereist dat de behandeling van het PSET-kommando terdege is bestudeerd.

In de meest eenvoudige vorm kunnen we een cirkel laten tekenen door de locatie van het middelpunt van de cirkel en een straal te specificeren. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 CIRCLE (111,111),50
30 GOTO 30
RUN
```

Een cirkel wordt getekend met het middelpunt op (111,111) en een straal van 50 puntjes. In feite wordt geen zuivere cirkel maar een ellips getekend. Dit is het gevolg van het feit dat de puntjes vertikaal 'dichter op elkaar' zitten dan horizontaal.

De straal dient groter dan of gelijk aan nul te zijn. Daarbij mag de straal niet groter zijn dan 32767. Indien de straal bestaat uit een gebroken waarde, dan wordt deze waarde eerst ontdaan van de decimale fractie.

Een foutje in MSX-basic draagt er zorg voor dan een negatieve straal, mits groter dan of gelijk aan –32768, wel wordt geaccepteerd. De computer tekent in zo'n geval de cirkel echter niet; in plaats hiervan blijft de computer op het CIRCLE-kommando 'hangen' en kan *alleen met CONTROL-STOP* worden onderbroken.

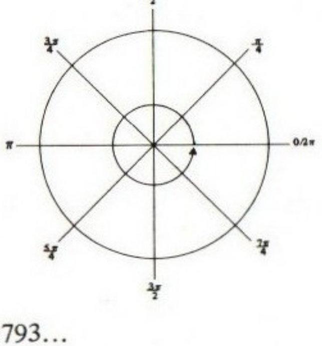
In plaats van de actieve voorgrondkleur kunnen we een andere voorgrondkleur specificeren. Bijvoorbeeld:

```
NEW
10 COLOR ,0,0
20 SCREEN 2
30 CIRCLE (111,111),50,12
40 GOTO 40
RUN
```

Een donkergroene cirkel wordt op een zwarte achtergrond getekend.

Een aanvangs- en een eindhoek kan worden gespecificeerd. De cirkel wordt dan gedeeltelijk getekend. Om precies te begrijpen welk deel van de cirkel gaat worden getekend, is een klein stukje goniometrie noodzakelijk.

De begin- en eindhoek dienen in radialen te worden opgegeven. De betekenis van de beginhoek en de eindhoek moge blijken uit het volgende schema:



$\pi = 3.141592653589793\ldots$

Om het volgende cirkeldeel te tekenen:



dient een beginhoek van $\frac{\pi}{4}$ en een eindhoek van $\frac{7\pi}{4}$ te worden opgenomen. $\frac{\pi}{4}$ is ongeveer gelijk aan 0.7854 en $\frac{7\pi}{4}$ is ongeveer gelijk aan 5.4978. Om dit cirkeldeel van een bepaalde cirkel te tekenen, kunnen we dus programmeren:

```
NEW
10 SCREEN 2
20 CIRCLE (111,111),50,,0.7854,5.4978
30 GOTO 30
RUN
```

Het betreffende cirkeldeel wordt getekend.

Indien we verbindingen naar het middelpunt willen tekenen, dienen we de begin- en eindhoek negatief op te geven. Bijvoorbeeld:

```
20 CIRCLE (111,111),50,,-0.7854,5.4978
30 GOTO 30
RUN
```

Het cirkeldeel verschijnt; de verbindingen met het middelpunt zijn aangebracht.

Probeer het bovenstaande voorbeeld ook eens met een SCREEN 1 kommando; precies dezelfde cirkel wordt, zij het in grovere punten, getekend.

Tenslotte kunnen we ook nog een afplattingsfaktor definiëren. Door deze faktor kunnen we in plaats van een cirkel een ellips tekenen. Indien de afplattingsfaktor groter is dan de waarde 1, dan wordt in

vertikale richting de straal op de juiste maat gehouden en wordt de straal in horizontale richting gedeeld door deze afplattingsfaktor. Het resultaat is dus een vertikaal gerichte ellips.
Indien de afplattingsfaktor kleiner is dan de waarde 1, dan wordt de straal in horizontale richting op de juiste maat gehouden en wordt de vertikale straal vermenigvuldigd met de afplattingsfaktor. Het resultaat is dus een horizontaal gerichte ellips. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 FOR P=.2 TO 5 STEP .2
30 CIRCLE (111,111),50,,,,P
40 NEXT P
50 GOTO 50
RUN
```

Op het beeldscherm verschijnt een samenspel van ellipsen.

N.B.: De afplattingsfaktor moet altijd groter zijn dan nul.

De cirkel behoeft niet volledig binnen het beeldschermbereik te liggen; hij mag zelfs volledig buiten het beeldschermbereik worden getekend. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 CIRCLE (-111,-111),200
30 GOTO 30
RUN
```

Een klein gedeelte van de cirkel, het gedeelte dat binnen het beeldschermbereik valt, wordt afgebeeld.

Uit het volgende voorbeeld blijkt dat indien een ellips of cirkel(deel) wordt getekend, de computer aanneemt dat het laatst getekende punt wordt gevormd door het middelpunt:

```
NEW
10 SCREEN 2
20 CIRCLE (111,111),50
30 LINE -STEP(100,100)
40 GOTO 40
RUN
```

Een cirkel met een vanuit het middelpunt naar rechtsonder lopende lijn is het resultaat. Deze lijn vangt aan in het laatst getekende punt; in dit geval het middelpunt (alhoewel dit niet daadwerkelijk is getekend).

SLEUTELWOORD CLEAR

soort .... **KOMMANDO**

schrijfwijze

```
CLEAR[<STRINGRUIMTE>[,<BOVENGRENS GEHEU-
   GEN>]]
<STRINGRUIMTE>::=<N>
<BOVENGRENS GEHEUGEN>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Eenvoudige betekenis

Met het enkele CLEAR-kommando worden de volgende acties ondernomen:

- alle eventueel nog openstaande kanalen worden gesloten (zelfde funktie als het CLOSE-kommando; zie aldaar)
- alle variabelen worden gewist
- alle array-variabelen worden van hun bijbehorende dimensies ontdaan (het DIM-kommando wordt als het ware voor elke variabele ongedaan gemaakt)

Bijvoorbeeld:

```
NEW
10 LET A$="DIT IS EEN TEST"
20 DIM B(20,20)
30 LET B(19,19)=22
40 CLEAR
50 PRINT A$;B(19,19)
RUN
Subscript out of range in 50
Ok
```

Uit bovenstaand programma blijkt dat de dimensionering voor variabele B is tenietgedaan. Hierdoor volgt een foutmelding. Tevens blijkt dat variabele A$ leeg is na uitvoering van CLEAR.

Gevorderd gebruik

Achter het CLEAR-kommando kunnen twee gegevens worden gespecificeerd. Om deze gegevens te vergelijken is het raadzaam om de geheugenindelingstabellen van hoofdstuk 1.3 nog eens te bestuderen.
Met het eerste gegeven kunnen we de maximaal door MSX te gebruiken string-ruimte bepalen. Indien we deze bepaling niet stellen, wordt een standaard grootte van 200 karakters aangenomen. Het onderstaande programma loopt bij een zojuist opgestarte MSX-computer fout:

```
NEW
10 FOR I=1 TO 255:A$=A$+"*":NEXT
RUN
Out of string space in 10
Ok
```

Deze foutmelding volgt uit het feit dat er standaard slechts een stringgebied van 200 karakters is toegewezen. Na intoetsen van:

```
CLEAR 1000:RUN
```

zal het programma nu wel tot een goed einde komen; er werd een string-gebied van 1000 karakters toegewezen.

Belangrijk is het om te weten dat MSX-basic bij het uitwerken van alfanumerieke uitdrukkingen een bepaalde werkruimte nodig heeft. Deze werkruimte is net zo groot als de grootste stringlengte die tijdens de uitwerking ontstaat. Hierdoor kan de bovengenoemde foutmelding eerder dan verwacht optreden. In het laatste voorbeeld trad deze reeds op bij de stringlengte van 100 tekens voor A$.

Met het tweede gegeven achter CLEAR kunnen we het laatste door MSX-basic te gebruiken geheugenadres bepalen. Hierdoor (zie geheugentabel) kan een vrije geheugenruimte worden gecreëerd waarin de specialist of zéér ver gevorderde amateur zijn of haar machinekode-programmatuur kan opslaan. Bijvoorbeeld:

```
CLEAR 200,5000
```

bepaalt dat het MSX-basic niet verder mag gaan in geheugengebruik dan geheugenadres 50000 en dat de stringruimte maximaal 200 karakters kan bevatten.

SLEUTELWOORD **CLOAD**

---

soort . . . . **KOMMANDO**

schrijfwijze

```
CLOAD[?][<BESTANDSNAAM>][,<...>]
```

betekenis

Zie eerst de behandeling van het CSAVE-kommando

Met het CLOAD-kommando kunnen programma's die met CSAVE op cassetteband zijn vastgelegd, weer worden ingelezen in het computergeheugen. Indien het vraagteken is opgenomen, wordt een programma niet van cassette geladen doch wordt slechts gecontroleerd of een programma op band overeen komt met het in het computergeheugen opgeslagen programma.
Indien geen programmanaam wordt gespecificeerd, wordt het eerst op de cassetteband voorkomende programma ingelezen. Indien wel een programmanaam is gespecificeerd, dan wordt de band afgezocht naar

het juiste programma en wordt alleen dit programma in behandeling genomen. Bijvoorbeeld:

```
NEW
10 REM *********************
20 REM TESTPROGRAMMA
30 REM *********************
40 STOP
```

(nu eerst een cassetteband plaatsen en de cassetterecorder op opnemen zetten)

`CSAVE "TEST",2` (met een schrijfsnelheid van 2400 baud wordt het programma op cassetteband geschreven)

(nu de cassetterecorder stoppen)

```
CLOAD? "TEST"
```

(nu de cassette terugspoelen en de recorder op afspelen zetten)

```
FOUND:TEST
Ok
```

(het programma is *vergeleken* en goed bevonden. Indien de vergelijking fout gaat, wordt de foutmelding Verify error gegeven)

(nu de computer uitschakelen en weer inschakelen; het programma is gegarandeerd uit het geheugen verdwenen.)

```
CLOAD
```

(nu de band terugspoelen en de recorder op afspelen zetten)

```
FOUND:TEST
Ok
LIST
10 REM *********************
20 REM TESTPROGRAMMA
30 REM *********************
40 STOP
Ok
```

(het programma is weer ingelezen)

Merk op dat bij de laatste keer geen programmanaam werd opgegeven; het eerste op de band voorkomende programma werd geladen.

Het is zéér raadzaam om *altijd* een met CSAVE vastgelegd programma te controleren op juistheid met een CLOAD?-kommando, direkt daarna uitgevoerd. De cassetteband is geen erg betrouwbaar opslagmedium; een eventuele 'drop-out' op een cassetteband kan vele uren programmeerwerk te niet doen!

Indien een programma dient te worden ingelezen terwijl we niet zeker weten of het het eerste programma op de band is, dienen we een programmanaam op te geven teneinde zekerheid te hebben dat het juiste programma wordt geladen. Stel voor dat op een cassetteband twee programma's TEST1 en TEST2 achter elkaar staan en we TEST2 willen laden. We kunnen dan de band terugspelen, de recorder op afspelen zetten en dan ingeven:

```
CLOAD "TEST2"
SKIP:TEST1
FOUND:TEST2
Ok
```

Bij SKIP (= overslaan) vermeldt de computer dat hij een programma wel heeft gevonden maar niet inleest. Bij FOUND (gevonden) geeft de computer aan dat hij inleest.

MSX schrijft geen controle voor op de integriteit (betrouwbaarheid) van ingelezen gegevens. Een verkeerd ingesteld volume of een kwalitatief slechte apparatuur kan leiden tot verminkte programma's bij inlezen zonder dat daarvan tijdens het inlezen melding wordt gemaakt. Zet het volume bij afspelen altijd op ongeveer 80 procent en kies bij de CSAVE voor de laagste schrijfsnelheid indien u de apparatuur niet geheel vertrouwt.

Een CLOAD behoeft geen snelheidsindicatie te bevatten zoals een CSAVE. De CLOAD bepaalt de snelheid waarmee werd geschreven automatisch en leest ook weer op deze snelheid in.

Het CLOAD-kommando, opgenomen in een programmaregel, heeft tot gevolg dat het betreffende programma eerst wordt ingelezen en

dat daarna de Ok-melding verschijnt (het programma loopt dus niet door).

N.B.: Het schrijven naar of laden van band van een groot programma kan tot verscheidene minuten duren.

Merk op dat het CLOAD-kommando standaard onder funktietoets 7 aanwezig is.

SLEUTELWOORD **CLS**

---

soort . . . . **KOMMANDO**

schrijfwijze

```
CLS
```

betekenis

Uitvoering van dit kommando resulteert in het geheel schoonvegen van het beeldscherm. Ook indien een grafisch beeldscherm geactiveerd is, zal dit volledig worden schoongemaakt. De grafische pointer (het 'laatst getekende punt') wordt dan niet veranderd.
De cursor wordt indien aanwezig, na het schoonmaken van het beeldscherm op de linker bovenpositie van het beeldscherm geplaatst. Voorbeeld:

```
NEW
10 INPUT "HOE HEET U ";NAAM$
20 CLS
30 PRINT "HALLO ";NAAM$
40 STOP
```

– Bij het uitvoeren van dit programma vraagt de computer eerst om uw naam. Daarna wordt het beeldscherm schoongemaakt en wordt de tekst HALLO, gevolgd door uw naam, afgedrukt.

SLEUTELWOORD # COLOR

soort .... **KOMMANDO**

schrijfwijze

```
COLOR[<VOORGROND>][,[<ACHTERGROND>][,
  <RAND>]]
```

betekenis

Met dit kommando kan de kleurinstelling van het beeldscherm worden bepaald. Een voorgrondkleur (de kleur van de letters of tekening), een achtergrondkleur (de kleur van het scherm) en een randkleur (de kleur van de boven- en onderrand van het beeldscherm) kan worden gespecificeerd.

De volgende kleuren kunnen worden gekodeerd:

kleurnummer	kleur of effect
0	doorschijnend
1	zwart
2	groen
3	lichtgroen
4	donkerblauw
5	lichtblauw
6	donkerrood
7	cyaan (een soort blauw)
8	rood
9	lichtrood
10	donkergeel
11	lichtgeel
12	donkergroen
13	magenta (een soort paars)
14	grijs
15	wit

Kleurkode 0 (transparant) resulteert in zwart indien toegepast als achtergrondkleur of randkleur. Toegepast als voorgrondkleur heeft kleurkode 0 tot gevolg dat de diverse afbeeldingen (tekst of tekeningen) niet zichtbaar zijn.

`COLOR 3,1,2`	de voorgrondkleur werd lichtgroen gekozen. De achtergrond is zwart en de rand is groen
`COLOR ,,14`	alleen de randkleur is naar grijs veranderd
`COLOR 0`	de voorgrond is nu doorschijnend en dus onzichtbaar
`COLOR ,0,0`	de achtergrond en rand zijn zwart (doorschijnend) gekleurd; de voorgrondkleur is ongewijzigd
`COLOR 15,4,4`	de standaard kleurinstelling (wit op donkerblauw, rand ook donkerblauw). Deze funktie is standaard onder funktietoets 6 aanwezig.

Merk op dat het COLOR sleutelwoord onder funktietoets nummer 1 aanwezig is.

Indien de numerieke uitdrukkingen die de kleuren aangeven, gebroken waarden bevatten, worden deze waarden ontdaan van de decimale fractie.
De kleurwaarden mogen niet kleiner zijn dan 0 en niet groter dan 15.

Tenslotte merken we op dat een COLOR-kommando vóór een SCREEN-kommando dient plaats te vinden teneinde de bedoelde achtergrondkleur en randkleur in een grafische instelling van het beeldscherm te realiseren. Indien een COLOR na een SCREEN-kommando wordt uitgevoerd, blijft de *oude* achtergrond- en randkleur actief.

SLEUTELWOORD # CONT

---

soort . . . . **KOMMANDO**

schrijfwijze

<u>CONT</u><...>

betekenis

Met CONT kunnen we een via het STOP-kommando of via de CONTROL-STOP afgebroken programma weer hervatten. Voorbeeld:

```
NEW
10 PRINT "REGEL 1"
20 STOP
30 PRINT "REGEL 2"
40 END
RUN
REGEL 1
Break in 20
CONT
REGEL 2
Ok
```

Indien een programma door middel van een STOP werd onderbroken, begint de hervatting van het programma na CONT bij de volgende programmaregel. Indien het programma echter door CONTROL-STOP werd onderbroken, dan begint de hervatting bij het kommando waarin het programma werd onderbroken. Voorbeeld:

Indien na onderbreking reeds wijzigingen in het programma werden aangebracht, zal de computer melden: Can't CONTINUE. Het is voor de computer dan niet mogelijk om de uitvoering te hervatten.

Merk op dat CONT standaard onder funktietoets 8 aanwezig is.

SLEUTELWOORD COS

---

soort . . . . **N-FUNKTIE**

schrijfwijze

```
COS(<HOEK>)
```

betekenis

Deze funktie geeft als resultaat de cosinus van de waarde van de tussen haakjes vermelde uitdrukking. Deze waarde wordt beschouwd als de uitdrukking van een hoek in radialen. Het resultaat ligt uiteraard altijd tussen –1 en 1.

Alhoewel elke waarde voor de hoek is toegestaan, is het in verband met de precisie van berekenen raadzaam om niet te grote hoekmaten te gebruiken.

Voorbeeld:

```
NEW
10 FOR I=0 TO 90
20 LET HOEK=I/180*3.1415926535
30 PRINT I;":";COS(HOEK)
40 NEXT I
RUN
```

(een cosinustabel verschijnt op het beeldscherm. Merk op dat op regel 20 de hoek van graden naar radialen wordt geconverteerd)

SLEUTELWOORD CSAVE

---

soort .... **KOMMANDO**

schrijfwijze

```
CSAVE<BESTANDSNAAM>[,<SCHRIJFSNELHEID>]
```

betekenis

Met het CSAVE-kommando kunnen programma's op cassetteband worden vastgelegd. Dit vastleggen gebeurt gekodeerd; het programma wordt op cassetteband vastgelegd in de kodes zoals het zich in het geheugen bevindt; dit in tegenstelling tot het SAVE-kommando waarin het programma in tekstuele vorm wordt vastgelegd.

Programma's, vastgelegd met CSAVE, kunnen met een RUN-kommando niet automatisch worden opgestart; hiertoe dient gebruik te worden gemaakt van het SAVE-kommando.

Met het CSAVE-kommando dient de programmanaam op cassette te worden bepaald. Deze naam MOET minimaal één teken en MAG maximaal 6 tekens bevatten. Een teveel aan tekens wordt verwaarloosd.

Als tweede gegeven bij de CSAVE kan eventueel een schrijfsnelheid worden gespecificeerd. De numerieke uitdrukking die de schrijfsnelheid bepaalt, mag gelijk zijn aan 1 of 2. Indien deze numerieke uitdrukking een gebroken waarde bevat, dan wordt een decimale fractie verwaarloosd. De snelheidsaanduiding heeft de volgende betekenis:

1: er wordt met 1200 baud geschreven, hetgeen wil zeggen dat er 1200 bits (ongeveer 150 karakters) per seconde worden overgedragen op de band
2: er wordt met 2400 baud geschreven; de dubbele snelheid dus.

Indien geen schrijfsnelheid wordt gespecificeerd, wordt automatisch een snelheid van 1200 baud aangenomen of de laatste (via CSAVE of SCREEN) bepaalde schrijfsnelheid indien deze werd gespecificeerd.

Indien men over een cassetterecorder van goede kwaliteit beschikt, en daarbij kwalitatief goede cassettebanden gebruikt, kan voor de hoogste snelheid worden gekozen. Echter, wanneer de kwaliteiten wat lager liggen, is het zéér raadzaam om de laagste snelheid te kiezen.

MSX schrijft geen lees/schrijfcontrole voor; een slechte kwaliteit van apparatuur kan veroorzaken dat er fouten optreden tijdens het vastleggen.

voorbeeld

Zie het voorbeeld bij CLOAD

SLEUTELWOORD CSRLIN

soort . . . . **SYSTEEMVARIABELE**

schrijfwijze

```
CSRLIN
```

betekenis

Deze systeemvariabele geeft als resultaat het regelnummer waarop de cursor zich op het moment bevindt. Voorbeeld:

```
NEW
10 FOR I=0 TO 5:LOCATE ,I
20 PRINT CSRLIN:NEXT:STOP
RUN
```

(In de linkerbovenhoek van het beeld verschijnen onder elkaar de cijfers 0 tot en met 5.)

De systeemvariabele CSRLIN kan geen waarde worden toegekend; de waarde kan alleen worden gebruikt.

SLEUTELWOORD DATA/READ/RESTORE

soort . . . . **KOMMANDO**

schrijfwijze

```
DATA{,}<DATA-ELEMENT>[,{,}<DATA-ELEMENT>]
READ<VARIABELE>{,<VARIABELE>}
RESTORE[<REGELNUMMER>]
```

betekenis

Behalve het sleutelwoord DATA worden ook de sleutelwoorden READ en RESTORE hier behandeld.

Met het DATA-sleutelwoord kunnen we vaste gegevens in een programma opnemen. Deze gegevens mogen bestaan uit numerieke konstanten of alfanumerieke konstanten. Bijvoorbeeld:

```
10 DATA 1.2,3.4,123.6,-12.33,"MAANDAG",
"PIETJE"
```

De diverse gegevens dienen door middel van een komma van elkaar te worden gescheiden.

De op deze wijze in een programma opgenomen gegevens kunnen door middel van een READ-kommando in een variabele worden ingelezen. Bijvoorbeeld:

```
NEW
10 DATA "MAANDAG","DINSDAG","WOENSDAG"
20 DATA "DONDERDAG","VRIJDAG","ZATERDAG"
30 DATA "ZONDAG"
40 FOR I=1 TO 7:READ DAG$
50 PRINT DAG$:NEXT I:STOP
RUN
MAANDAG
DINSDAG
WOENSDAG
DONDERDAG
VRIJDAG
ZATERDAG
ZONDAG
Break in 50
Ok
```

In het laatste voorbeeld worden de alfanumerieke konstanten MAANDAG...VRIJDAG achtereenvolgens in de variabele DAG$ ingelezen en vervolgens afgedrukt op het beeldscherm. Bij een READ 'onthoudt' de computer waar hij is gebleven met inlezen. Is MAANDAG reeds ingelezen, dan zal de volgende keer dat een READ wordt ingelezen, automatisch DINSDAG worden ingelezen. Het maakt hiervoor niet uit wáár precies de konstanten in een DATA-commando zijn genoemd; de computer begint altijd bij de eerste konstante en werkt deze konstante in volgorde van voorkomen af. Bijvoorbeeld:

```
NEW
10 DATA 33,44,55,66
20 READ A:PRINT A:GOTO 20
30 DATA 77,88,99
RUN
 33
 44
 55
 66
 77
 88
 99
Out of DATA in 20
Ok
```

We zien dat alle genoemde konstanten worden ingelezen en afgedrukt. Het maakt niet uit of de gegevens in een DATA-kommando voor of na de leesopdracht zijn genoemd. Pas als er geen volgende gegevens meer kunnen worden gevonden, geeft de computer een foutmelding.

Met een RESTORE-kommando kunnen we de computer opdracht geven, bij de volgende READ-opdracht weer bij een bepaald gegeven te beginnen met inlezen. Bijvoorbeeld:

```
NEW
10 DATA 1,2,3,4,5
20 RESTORE 20
30 DATA 111,222,333
40 READ A,B,C
50 PRINT A;B;C
RUN
 111  222  333
Ok
```

Met de RESTORE 20 opdracht gaven we in bovenstaand voorbeeld aan dat het inlezen op of na programmaregel 20 diende te geschieden. Alleen een RESTORE-opdracht (dus zonder regelnummer) heeft tot gevolg dat de computer weer bij het allereerste DATA-gegeven begint.

Bijvoorbeeld:

```
NEW
10 READ A:IF A=0 THEN RESTORE:GOTO 10
20 PRINT A;:READ A$:PRINT A$:GOTO 10
1000 DATA 1,"ZONDAG",2,"MAANDAG",
3,"DINSDAG"
1010 DATA 4,"WOENSDAG",5,"DONDERDAG",6
1020 DATA "VRIJDAG",7,"ZATERDAG",0
RUN
 1 ZONDAG
 2 MAANDAG
 3 DINSDAG
 4 WOENSDAG
 5 DONDERDAG
 6 VRIJDAG
 7 ZATERDAG
 1 ZONDAG
 2 MAANDAG...
```

(etcetera, de dagen van de week blijven op het beeldscherm voorbijkomen. Onderbreken met CONTROL-STOP)

In bovenstaand voorbeeld wordt steeds een dagnummer en een dagnaam ingelezen; in variabele A en A$. Indien op regel 10 echter voor variabele A een nul-waarde wordt ingelezen, wordt een RESTORE uitgevoerd waarna opnieuw de READ wordt uitgevoerd. Na deze RESTORE wordt op regel 10 weer de waarde 1 voor variabele A ingelezen en wordt op regel 20 de waarde ZONDAG voor variabele A$ ingelezen.

Merk op dat numerieke gegevens in een DATA-regel wel met READ in een alfanumerieke variabele kunnen worden gelezen, maar dat alfanumerieke gegevens niet in een numerieke variabele kunnen worden gelezen; dit laatste resulteert in een Sytax error op de betreffende DATA-regel.

SLEUTELWOORD

DEFSTR / DEFINT DEFSNG / DEFDBL

soort .... **KOMMANDO**

schrijfwijze

```
DEFSTR<LETTER>[-<LETTER>]{,<LETTER>
  [-<LETTER>]}
DEFINT<LETTER>[-<LETTER>]{,<LETTER>
  [-<LETTER>]}
DEFSNG<LETTER>[-<LETTER>]{,<LETTER>
  [-<LETTER>]}
DEFDBL<LETTER>[-<LETTER>]{,<LETTER>
  [-<LETTER>]}
```

betekenis

Voor een goed begrip van onderstaande funkties is het raadzaam, de hoofdstukken 5 en 6 nog eens door te nemen.

In MSX-basic wordt een variabele, indien de naam niet één van de achtervoegsels $, %, ! of # heeft, automatisch gemaakt tot een variabele met dubbele precisie. Indien we bijvoorbeeld programmeren:

```
NEW
10 LET A=1/3
20 PRINT A
RUN
 .33333333333333
Ok
```

Dan zien we dat voor de variabele A automatisch een numerieke variabele werd gekozen met een precisie van 14 cijfers; een dubbele precisie dus. Als het ware werd automatisch voor het achtervoegsel #gekozen.

Dit automatisme kunnen we sturen door gebruik van de DEFSTR, de DEFINT, de DEFSNG en de DEFDBL. Achter het betreffende sleutel-

woord dienen letterreeksen te worden aangegeven. Na het uitvoeren van één van de genoemde kommando's wordt voortaan voor een variabele met een beginletter, genoemd in een letterreeks, een ander variabeletype gehanteerd (indien het type niet via een achtervoegsel is bepaald).

De sleutelwoorden hebben het volgende effekt:

DEFSTR	als standaard variabele-type wordt het type alfanumerieke variabele gehanteerd
DEFINT	als standaard variabele-type wordt het type integere variabele gehanteerd
DEFSNG	als standaard variabele-type wordt het type variabele met enkelvoudige precisie gehanteerd
DEFDBL	als standaard variabele-type wordt het type variabele met dubbele precisie gehanteerd

De letterreeksen die achter een DEFSTR, DEFINT, DEFSNG of DEFDBL worden opgenomen, kunnen uit twee soorten bestaan, namelijk de enkelvoudige opsomming en de reeksaanduiding. Bij de enkelvoudige opsomming worden alle geldende letters, gescheiden door een komma, opgegeven. Bij de reeksaanduiding wordt de eerste geldende letter, een min-teken en de laatste geldende letter opgegeven. Bijvoorbeeld:

DEFSTR A,C,F	alle variabelen waarvan de naam begint met een A, een C of een F en geen achtervoegsel hebben ($, %, ! of #), worden **automatisch** beschouwd als alfanumerieke variabelen (krijgen als het ware automatisch een $ als achtervoegsel)
DEFSNG C-Q,Z	alle variabelen waarvan de naam begint met een C, D, E...Q of met een Z en daarbij geen achtervoegsel hebben, worden automatisch beschouwd als variabelen met enkelvoudige precisie (krijgen als het ware automatisch een ! als achtervoegsel)

Voorbeeld:

```
NEW
10 DEFSTR A-C
20 A="ALFANU"
30 B="MERIEKE V"
40 C="ARIABELEN"
50 PRINT A;B;C
55 PRINT A$;B$;C$
60 A%=32000:PRINT A%:STOP
RUN
ALFANUMERIEKE VARIABELEN
ALFANUMERIEKE VARIABELEN
 32000
Break in 60
Ok
```

We zien dat de variabelen A, B, C, indien niet met achtervoegsel gebruikt, na de DEFSTR A–C automatisch als alfanumerieke variabelen worden beschouwd. Op programmaregel 60 zien we echter dat wanneer een achtervoegsel wordt gebruikt, dit achtervoegsel geldend is. Ook zien we dat het achtervoegsel (regel 55) eventueel wel mag worden gebruikt.

Merk op dat een reeksaanduiding waarbij de tweede letter kleiner is dan de eerste letter (bijvoorbeeld een DEFDBL C–A) een syntax error (fout in schrijfwijze) veroorzaakt.

SLEUTELWOORD # DELETE

---

soort . . . . **KOMMANDO**

schrijfwijze

```
DELETE[<REGELNUMMER>][-<REGELNUMMER>]
```

betekenis

Met dit kommando kunt u:

- Een regel uit het programma verwijderen. Geef in dat geval DELETE in, gevolgd door het regelnummer of een punt voor het laatst behandelde regelnummer.
- Een eerste gedeelte van een programma verwijderen. Geef hiertoe DELETE in, gevolgd door een minteken en het laatste te verwijderen regelnummer.
- Een laatste gedeelte van een programma verwijderen. Geef hiertoe DELETE in, gevolgd door het eerste te verwijderen regelnummer en een minteken.
- Een middengedeelte van een programma verwijderen. Geef hiertoe DELETE in, gevolgd door het eerste te verwijderen regelnummer, een minteken en het laatste te verwijderen regelnummer.

Voorbeelden:

```
DELETE 60
```

regel 60 wordt verwijderd

```
DELETE 60-120
```

regels 60 tot en met 120 worden verwijderd

```
DELETE -90
```

alle regels tot en met 90 worden verwijderd

```
DELETE .
```

de laatste behandelde regel wordt verwijderd

De aangegeven regelnummers dienen bestaande regelnummers te zijn.

Indien DELETE in een programmaregel is opgenomen, worden de daarbij vermelde regelnummers meehernummerd (zie de behandeling van RENUM).

SLEUTELWOORD **DIM**

soort .... **KOMMANDO**

schrijfwijze

```
DIM<RIJ VAN ARRAY-VARIABELEN>
```

betekenis

Met het DIM-kommando kunnen we zogenaamde array-variabelen toewijzen. Een array-variabele is te vergelijken met een tabel waarin we meerdere waarden kunnen opslaan.
Zowel het aantal dimensies als het aantal elementen per dimensie is praktisch onbeperkt. Theoretisch is het maximum aantal dimensies gelijk aan 255 en het maximum aantal elementen per dimensie gelijk aan 65535. Indien tussen haakjes een numerieke uitdrukking voorkomt met een gebroken waarde, dan wordt een decimale fractie verwaarloosd. Een array kan één-dimensionaal (een rij van waarden), twee-dimensionaal (een matrix van waarden), drie-dimensionaal (een ruimtelijke matrix) maar ook méér-dimensionaal zijn. Alhoewel het werken met meerdere dimensies een abstract denkvermogen vereist, kan dit toch wel eens erg gemakkelijk zijn.

Voorbeeld:

```
NEW
10 DIM A(3,3):LET T=0
20 FOR I=0 TO 3:FOR J=0 TO 3
30 LET A(I,J)=T:LET T=T+1
40 NEXT J,I
50 INPUT "RIJ ";R:IF NOT(R) THEN STOP
60 INPUT "KOLOM ";K
70 PRINT "DE WAARDE OP";R;",";K
80 PRINT "IS";A(R,K):GOTO 60
RUN
RIJ ? 2
KOLOM ? 2
DE WAARDE OP 2 , 2
IS 10
```

```
RIJ ? 3
KOLOM ? 1
DE WAARDE OP 3 , 1
IS 13
RIJ ? 0
Break in 50
Ok
```

Indien een variabele-naam in een dim-kommando wordt genoemd zonder dimensiebepaling, dan wordt deze variabele slechts toegewezen en op nul gesteld.

Merk op dat meer arrays in een dim-kommando kunnen worden gedimensioneerd. Bijvoorbeeld:

```
10 DIM A(3,2),B$(2),C(1,2,5),B%(100,5)
```

Ook alfanumerieke tabellen kunnen worden aangelegd. Elk element in een alfanumerieke tabel mag een andere lengte hebben. Bijvoorbeeld:

```
NEW
10 DIM NAAM$(100)
20 INPUT "NUMMER ";NUMMER%
30 IF NUMMER%<0 OR NUMMER%>100 GOTO 20
40 PRINT "OUDE NAAM: ";NAAM$(NUMMER%)
50 INPUT "NIEUWE NAAM ";NAAM$(NUMMER%)
60 GOTO 20
RUN
NUMMER ? 1
OUDE NAAM:
NIEUWE NAAM ? FERDINAND
NUMMER ? 55
OUDE NAAM:
NIEUWE NAAM ? KAREL DE GROTE
NUMMER ? 1
OUDE NAAM: FERDINAND
NIEUWE NAAM ? FREDERIK
NUMMER ? 55
```

```
OUDE NAAM: KAREL DE GROTE
NIEUWE NAAM ?
```

SLEUTELWOORD DRAW

soort . . . . **KOMMANDO**

schrijfwijze

```
DRAW<TEKENAANWIJZING>
```

betekenis

Voor goed begrip van dit kommando is het noodzakelijk dat de PSET terdege is bestudeerd.

Met het DRAW-kommando kunnen tekeningen op het beeldscherm worden gemaakt. Achter het DRAW-kommando dient dan een alfanumerieke uitdrukking te worden opgenomen die de teken-aanwijzingen voor de computer bevatten. Deze alfanumerieke uitdrukking dient te zijn opgesteld volgens de GML (Graphics Macro Language) richtlijnen. GML kan als een eenvoudige, aparte taal worden gezien die door Microsoft is ontwikkeld en alleen een grafische toepassing heeft.

Binnen GML kennen we volgende, éénletterige kommando's:

M...,... de M staat voor MOVE. De getallen, achter MOVE opgenomen, geven aan naar welk punt moet worden gegaan op het beeldscherm. De geadresseerde beeldschermlocatie mag zich buiten het scherm bevinden. Vanuit het laatst getekende punt wordt een lijn getrokken naar het aangegeven punt. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 DRAW "M100,100M200,100M20,30M22,33"
30 GOTO 30
RUN
```

Een lijnenspel wordt getekend.
Indien voor de achter M opgenomen getallen een plus- of minteken staat, wordt de verplaatsing relatief beschouwd ten opzichte van het laatst getekende punt. Een M+100,100 heeft dan bijvoorbeeld tot gevolg dat een lijn vanuit het laatst getekende punt wordt getrokken naar een punt dat op de vertikale locatie 100 ligt en op een horizontale locatie, 100 beeldschermpunten verder naar rechts dan die van het laatst getekende punt.

Voorbeeld:

```
NEW
10 SCREEN 2
20 DRAW "M100,100M+11,0M0,-11M-11,0M0,
+11"
30 GOTO 30
RUN
```

Eerst wordt een lijn vanuit het laatst getekende punt naar (100,100) getrokken. Vervolgens wordt een vierkantje getekend door achtereenvolgens een lijn van 11 beeldpunten naar rechts, naar beneden, naar links en weer naar boven te tekenen.

U... de U staat voor UP. Vanuit het laatst getekende punt wordt een lijn naar boven getrokken ter lengte van een aantal aangegeven beeldschermpunten.

R... de R staat voor RIGHT. Vanuit het laatst getekende punt wordt een lijn naar rechts getrokken ter lengte van het aantal aangegeven beeldschermpunten.

D... de D staat voor DOWN. Vanuit het laatst getekende punt wordt een lijn naar beneden getrokken ter lengte van het aantal aangegeven beeldschermpunten.

L... de L staat voor LEFT. Vanuit het laatst getekende punt wordt een lijn naar links getrokken ter lengte van het aantal aangegeven beeldschermpunten.

Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 DRAW "M100,100U20R20D20L20"
30 GOTO 30
RUN
```

Vanuit het laatst getekende punt wordt eerst een lijn naar (100,100) getrokken. Daarna wordt vanuit dit punt een vierkantje getekend; twintig beeldpunten naar boven, naar rechts, naar beneden en weer naar links.

E... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten hoger en verder naar rechts gelegen. Er wordt dus een lijn rechts schuin naar boven getekend.

F... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten lager en verder naar rechts gelegen. Er wordt dus een lijn rechts schuin naar beneden getekend.

G... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten lager en verder naar links gelegen. Er wordt dus een lijn links schuin naar beneden getekend.

H... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten hoger en verder naar links gelegen. Er wordt dus een lijn links schuin naar boven getekend.

Voorbeeld:

```
NEW
10 SCREEN 2
20 DRAW "M100,100E10F10G10H10"
30 GOTO 30
RUN
```

Vanuit het laatst getekende punt wordt eerst een lijn naar (100,100) getekend. Daarna wordt vanuit dit punt een ruitfiguur getekend; tien

beeldpunten naar rechtsboven, tien beeldpunten naar rechtsonder, tien beeldpunten naar linksonder en tien beeldpunten naar linksboven.

B... de B staat voor BLANK. De eerst volgende tekenopdracht wordt wel uitgevoerd maar de lijn wordt niet getrokken. In feite wordt met behulp van deze funktie een ander punt aangewezen als zijnde het laatst getekend. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 DRAW "BM100,100E10F10G10H10"
30 GOTO 30
RUN
```

Net als in het laatste voorbeeld wordt ook hier een ruitfiguur getekend. Echter, de eerste lijn (M100,100) die vanuit het laatst getekende punt naar (100,100) zou moeten lopen, wordt niet getekend.

N... de eerstvolgende tekenopdracht wordt niet van begin- naar eindpunt maar van eind- naar beginpunt uitgevoerd. Hierdoor blijft het laatst getekende punt vóór deze opdracht ook het laatst getekende punt ná deze opdracht. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 DRAW "BM100,100NU50ND50NL50NR50NE50NF
50NG50NH50"
30 GOTO 30
RUN
```

Steeds wordt vanuit punt (100,100) een lijn getrokken naar diverse richtingen waardoor een sterfiguur ontstaat.

A... de letter A staat voor ANGLE. Een hoek van 0 graden (A0), 90 graden (A1), 180 graden (A2) of 270 graden (A3) kan worden gespecificeerd. Alle volgende tekenopdrachten (tot een volgend A-kommando) worden het aantal aangegeven graden naar links gedraaid. Bijvoorbeeld:

```
NEW
10 LINE INPUT "HOEK:";H$
20 SCREEN 2
30 DRAW "A0BM100,100A"+H$+"D20R20U20L20
E10F10"
40 GOTO 40
RUN
HOEK:
```

Een hoek (0, 1, 2 of 3) kan worden ingegeven. Een eenvoudig tekeningetje van een huis wordt, gedraaid volgens de aangegeven hoek, getekend. RUN dit programma diverse malen met verschillende hoeken.

C... de C staat voor COLOR. Een voorgrondkleur (C0... C15) kan worden opgegeven. Indien geen kleur wordt gespecificeerd, wordt de op dat moment actieve voorgrondkleur als actieve kleur gebruikt. Een eenmaal gespecificeerde kleur blijft geldig totdat binnen het DRAW-kommando een volgende kleur wordt gespecificeerd. Bijvoorbeeld:

```
15 LINE INPUT "KLEUR:";C$
25 DRAW "C"+C$
RUN
HOEK:
```

Bij het vorige voorbeeld werden twee programmaregels toegevoegd. Nu kan ook een kleurkode worden ingegeven waarna het tekeningetje in de gewenste kleur wordt uitgevoerd.

S... de S staat voor SCALE. Een schaal kan worden opgegeven. Het achter de S opgenomen getal, gedeeld door vier, vormt het aantal werkelijk getekende puntjes ten opzichte van één geprogrammeerd puntje. Indien geen schaalfaktor wordt opgegeven, of een S0 of een S4 wordt opgegeven, komt een getekend puntje precies overeen met een geprogrammeerd puntje. Een S1 heeft tot gevolg dat een tekening vier maal zo klein wordt uitgevoerd terwijl een S16 juist tot gevolg heeft dat een tekening vier maal zo groot wordt uitgevoerd. Het M-kommando met absolute adressering (dus zonder gebruik van een plus- of minteken) wordt door het S-kommando als enige niet beïn-

vloed. Het S-kommando blijft geldig totdat een nieuw S-kommando wordt gegeven. Bijvoorbeeld:

```
16 LINE INPUT "SCHAAL:";S$
26 DRAW "S"+S$
RUN
HOEK:
```

Alleen regelnummers 16 en 26 werden aan het vorige voorbeeld toegevoegd. Nu kan ook een schaal worden opgegeven. Afhankelijk van de opgegeven schaal wordt het tekeningetje nu groter of kleiner afgebeeld.

X...; de tussen de X en de puntkomma vermelde alfanumerieke variabele wordt tijdens het tekenen tussengevoegd. Hierdoor kan een tekening die is gekodeerd in een aparte variabele, diverse malen worden herhaald. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 Q$="U10R10D10L10E10H5G5F10"
30 DRAW "S4C15A0BM99,99XQ$;"
40 GOTO 40
RUN
```

Een eenvoudig tekeningetje zoals in Q$ gekodeerd, wordt in regel 30 getekend nadat schaal, kleur, hoek en beginpunt zijn voorbereid.

=...; de tussen het gelijkteken en de puntkomma vermelde numerieke variabele wordt tijdens het tekenen als waarde ingevoegd. Hierdoor behoeft een string met een tekenopdracht niet moeizaam met behulp van onder meer het STR$-sleutelwoord worden opgebouwd maar kan onmiddellijk vanuit een numerieke waarde een afmeting of locatie worden bepaald. Bijvoorbeeld:

```
NEW
10 INPUT "GROOTTE ";GR%:GT%=GR%/2
20 SCREEN 2
30 DRAW "S4C15A0BM99,99U=GR%;R=GR%;D=GR%;
```

```
L=GR%;"
40 DRAW "E=GR%;H=GT%;G=GT%;F=GR%;"
50 GOTO 50
RUN
GROOTTE ?
```

Een grootte in aantal beeldschermpuntjes kan worden ingegeven. Een eenvoudig tekeningetje wordt op de aangegeven grootte gemaakt.

Maxima en minima:

- Voor numerieke variabelen die met behulp van het gebruik van een gelijkteken in een DRAW-string worden opgenomen geldt dat deze variabelen geheel in waarde dienen te zijn. Een eventueel aanwezige decimale fractie wordt genegeerd.
- Voor de in verband met het U, E, R, F, D, G, L, H en M gebruikte konstanten of variabelen gelden de volgende bepalingen:
  variabele: minimaal –32768, maximaal 32787
  konstante: minimaal –65535, maximaal 65535
  deze waarden worden modulo 32768 behandeld, dat wil zeggen: indien een waarde groter is dan of gelijk is aan 32768 dan wordt 32768 van deze waarde afgetrokken en wanneer een waarde kleiner is dan of gelijk aan –32768 dan wordt 32768 bij deze waarde opgeteld.
- In verband met het S-kommando geldt een minimum van 0 en een maximum van 255 voor de betreffende konstante of variabele.
- In verband met het A-kommando geldt een minimum van 0 en een maximum van 3 voor de betreffende konstante of variabelen.
- In verband met het C-kommando geldt een minimum van 0 en een maximum van 15 voor de betreffende variabele of konstante.

SLEUTELWOORD # END

---

soort . . . . **KOMMANDO**

schrijfwijze

<u>END</u>

betekenis

Met het END-kommando geven we aan dat het programma is beeindigd. Meestal is het opnemen van een END-kommando slechts een formaliteit. Voorbeeld:

```
NEW
10 PRINT "DIT IS HET EINDE"
20 END
RUN
DIT IS HET EINDE
Ok
```

SLEUTELWOORD EXP

---

soort .... **N-FUNKTIE**

schrijfwijze

```
EXP(<N>)
```

betekenis

Deze funktie geeft als resultaat de waarde *e* (2,7182818284...), verheven tot de macht die gevormd wordt door de waarde van de tussen haakjes vermelde uitdrukking.

Voorbeeld:

```
NEW
10 INPUT "WAARDE ";A
20 PRINT "E TOT DE MACHT A=";EXP(A)
30 GOTO 10
RUN
WAARDE ? 1
E TOT DE MACHT A= 2.7182818284588
```

```
WAARDE ? 200
E TOT DE MACHT A=
Overflow in 20
Ok
```

Merk op dat de maximale waarde van de numerieke uitdrukking ligt bij 145,06286085862 en de minimale waarde bij –75453,410912322.

SLEUTELWOORD **FIX**

---

soort . . . . **N-FUNKTIE**

schrijfwijze

```
FIX(<N>)
```

betekenis

Het resultaat van deze funktie is een waarde, gelijk aan de waarde van de tussen haakjes vermelde numerieke uitdrukking maar dan ontdaan van decimale cijfers. FIX geeft dus altijd een geheel getal.

Voorbeeld:

```
NEW
10 LET A=12.5:LET B=-12.5
20 PRINT FIX(A);FIX(B)
RUN
 12 -12
Ok
```

SLEUTELWOORD # FOR-TO-STEP-NEXT

soort . . . . **KOMMANDO**

schrijfwijze

```
FOR<NUMERIEKE VARIABELE>=<STARTWAARDE>TO
  <EINDWAARDE>[STEP<STAPWAARDE>]
NEXT[<NUMERIEKE VARIABELE>{,NUME-
  RIEKE VARIABELE>}]
```

betekenis

De FOR...TO...STEP combinatie is in samenwerking met het NEXT-kommando één van de krachtigste besturingsmogelijkheden van de computer.

Allereerst wijst FOR de startwaarde toe aan de genoemde variabele zoals een LET-kommando dat zou doen. Vervolgens vervolgt het programma op gebruikelijke wijze.
Wanneer nu vervolgens een bijbehorend NEXT-kommando wordt tegengekomen, dan wordt de betreffende variabele met de stapwaarde verhoogd. Hierna wordt gecontroleerd of de betreffende variabele ondertussen de eindwaarde niet is gepasseerd. Wanneer dit nog niet zo blijkt te zijn, wordt hervat met het kommando dat direkt op de FOR... TO...STEP volgt. Is de variabele deze eindwaarde echter wel gepasseerd, dan wordt het programma na de NEXT voortgezet.

Voorbeeld:

```
NEW
10 FOR I=1 TO 4 STEP 1.5
20 PRINT I
30 NEXT I
40 PRINT "KLAAR"
RUN
 1
 2.5
 4
KLAAR
Ok
```

De stapwaarde mag ook negatief zijn. Bijvoorbeeld:

```
NEW
10 FOR A=10 TO 0 STEP -1
20 PRINT A;
30 NEXT A
RUN
 10  9  8  7  6  5  4  3  2  1  0
Ok
```

Indien STEP wordt weggelaten, wordt automatisch een stapwaarde 1 aangenomen. Bijvoorbeeld:

```
NEW
10 FOR P=2 TO 5:PRINT P*P:NEXT:STOP
RUN
 4
 9
 16
 25
Break in 10
Ok
```

Uit het vorige voorbeeld blijkt dat bij de NEXT niet noodzakelijk een variabele behoeft te worden vermeld.

Het geheel vanaf FOR...TO...STEP tot en met NEXT noemt men wel de for-next loop (loop = lus). Twee of meer for-next loops mogen binnen elkaar plaatsvinden. Bijvoorbeeld:

```
NEW
10 FOR A=1 TO 3
20 FOR B=1 TO 3
30 PRINT A*B;
40 NEXT B:PRINT
50 NEXT A
RUN
 1  2  3
```

```
 2  4  6
 3  6  9
Ok
```

Kruisende loops zijn echter verboden.

GOED

```
NEW
10 FOR A=1 TO 5
20 FOR B=1 TO 5
.
.      (rest programma)
.
90 NEXT B
100 NEXT A
```

FOUT

verhoden

```
NEW
10 FOR A=1 TO 5
20 FOR B=1 TO 5
.
.
.
90 NEXT A
100 NEXT B
```

Twee of meer NEXT-kommando's direkt ná elkaar mogen door één kommando worden vervangen. Voorwaarde is dan wel dat het betreffende NEXT-kommando de variabelen in de juiste volgorde noemt. Regel 90 en 100 van het bovenstaande goede voorbeeld mogen dus vervangen worden door de ene programmaregel:

```
90 NEXT B,A
```

In tegenstelling tot de meeste andere BASIC-dialecten voert MSX-basic een for-next loop altijd minimaal één keer uit, hoe de startwaarde en de eindwaarde zich ook tot elkaar verhouden.

SLEUTELWOORD — **FRE**

---

soort . . . . **N-FUNKTIE**

schrijfwijze

<u>FRE</u>(<U>)

betekenis

Deze funktie geeft als resultaat een getal dat de nog vrije geheugenruimte uitdrukt. Met de FRE-funktie kan

– de vrije ruimte in het geheugen, met uitzondering van het string-geheugen, worden bepaald. Hiertoe dient de tussen haakjes op te nemen uitdrukking numeriek te zijn

– de vrije ruimte in het string-gebied worden bepaald. Hiertoe dient de tussen haakjes te vermelden uitdrukking alfanumeriek te zijn.

Bijvoorbeeld:

(zet eerst de computer uit en weer aan)

```
PRINT FRE(0)
 28815
Ok
PRINT FRE("")
 200
Ok
```

Uit het bovenstaande blijkt dat de computer een vrij geheugenruimte-gebied heeft van 28815 tekens en dat in het vrije string-gebied nog plaats is voor 200 karakters.

SLEUTELWOORD

GOSUB / RETURN

soort . . . . **KOMMANDO**

schrijfwijze

```
GOSUB<REGELNUMMER>
RETURN[<REGELNUMMER>]
```

betekenis

Het is noodzakelijk dat eerst GOTO wordt bestudeerd.
In eerste instantie werkt de GOSUB hetzelfde als de GOTO; het programma gaat verder met het genoemde regelnummer. Echter, bij GOSUB doet de computer nog meer. De computer 'onthoudt' namelijk het punt *waarvandaan* wordt gesprongen, het zogenaamde terugkeeradres in een stukje geheugen, de stack (stapel) genaamd. Wanneer later een RETURN-kommando wordt gegeven (RETURN = keer terug), dan haalt de computer dit regelnummer weer van die stack en gaat door met het kommando dat direkt komt na de GOSUB (dus één stap verder dan wat de stack aangeeft).

Het voordeel van een GOSUB is, dat gedeelten van programmatuur meerdere malen kunnen worden gebruikt op diverse plaatsen in het programma. Bijvoorbeeld:

```
NEW
10 INPUT "HOEVEEL IS 2+3 ";U
20 IF U<>5 THEN GOSUB 500:GOTO 10
30 INPUT "EN HOEVEEL IS 5+6 ";U
40 IF U<>11 THEN GOSUB 500:GOTO 30
50 PRINT "GOEDZO, KLAAR!":STOP
500 REM SUBROUTINE FOUTMELDING
510 PRINT "NEE HOOR, DAT IS NIET"
520 PRINT "GOED. NOG EEN KEER!"
530 RETURN
RUN
HOEVEEL IS 2+3 ? 4
NEE HOOR, DAT IS NIET
GOED. NOG EEN KEER!
HOEVEEL IS 2+3 ? 5
EN HOEVEEL IS 5+6 ? 12
NEE HOOR, DAT IS NIET
GOED. NOG EEN KEER!
EN HOEVEEL IS 5+6 ? 11
GOEDZO, KLAAR!
Break in 50
Ok
```

We zien dat het programmadeel vanaf regel 500 een keer vanuit regel 20 en een keer vanuit regel 40 werd aangeroepen. Het programmadeel waarin de foutmelding wordt afgedrukt, behoeft door de GOSUB-mogelijkheid maar één maal in het programma te worden opgenomen.

Een programmagedeelte dat via een GOSUB wordt aangeroepen, noemt men meestal een onderprogramma of een *subroutine*.

Een subroutine mag weer een andere subroutine aanroepen. In dat geval wordt het te onthouden terugkeeradres bovenop de stack geplaatst. Dit heeft tot gevolg dat een RETURN altijd een terugkeer veroorzaakt naar het gedeelte na de laatste GOSUB.
Voorbeeld:

```
NEW
10 GOSUB 100:PRINT "TERUG
  IN HOOFDPROGRAMMA"
20 STOP
100 GOSUB 500:PRINT "TERUG
  IN EERSTE SUBROUTINE"
110 RETURN
500 PRINT "TWEEDE SUBROUTI-
  NE":RETURN
RUN
TWEEDE SUBROUTINE
TERUG IN EERSTE SUBROUTINE
TERUG IN HOOFDPROGRAMMA
Break in 20
Ok
```

Ga na dat de regelnummers waar de computer achtereenvolgens mee bezig is, de nummers 10, 100, 500, 100, 110, 10 en 20 zijn.

Een goede BASIC-programmeur stelt zijn programma uit vele subroutines samen. In elke subroutine verzorgt hij of zij dan een compleet afgerond geheel. Uiteindelijk stelt deze programmeur dan een HOOFD-PROGRAMMA samen waarin al deze subroutines met GOSUB-kommando's worden aangesproken. Wanneer het hoofdprogramma, dat bijna alleen uit GOSUB's bestaat, daarbij ook nog met REM-kommando's goed van kommentaar is voorzien, ontstaat een programma dat ook bij grote afmetingen voor iedereen nog zeer goed te volgen is.

Het is een gouden regel dat een subroutine ALTIJD met een GOSUB moet worden aangesproken en ALTIJD met een RETURN moet worden verlaten.

Indien tijdens het samenstellen van een programma de situatie ontstaat waarbij een normale RETURN niet meer kan worden gebruikt, kan een RETURN, gevolgd door een regelnummer, worden gebruikt. Dit *bijzonder weinig fraaie* gebruik van RETURN werkt als een GOTO met dit verschil dat een terugkeeradres van de stack wordt gehaald zonder dat er iets mee wordt gedaan; het geheugen wordt niet overbelast terwijl eventueel volgende RETURN-kommando's goed blijven werken (naar de juiste aanroeplocaties terugkeren). Voorbeeld:

```
NEW
10 GOSUB 100:STOP
100 INPUT "WAARDE TUSSEN 1 EN 10 ";A
110IF A=INT(A) AND A>0 AND A<11 THEN RE
TURN
120 RETURN 10
RUN
WAARDE TUSSEN 1 EN 10 ? 2.5
WAARDE TUSSEN 1 EN 10 ? 3
Break in 10
Ok
```

(via een RETURN 10 wordt een verbetering geforceerd)

SLEUTELWOORD GOTO

---

soort .... **KOMMANDO**

schrijfwijze

```
GOTO
GO TO <REGELNUMMER>
```

betekenis

Met GOTO kan de programmaloop worden veranderd. Het regelnummer dat achter GOTO wordt vermeld, is het regelnummer waar de computer verder gaat. Een GOTO mag naar een eerder- of verdergelegen regelnummer verwijzen. Ook mag GOTO naar zichzelf verwijzen.

Voorbeeld:

```
NEW
10 GOTO 10
RUN
```

– Het programma staat in een eeuwigdurende lus en kan slechts met een CONTROL-STOP worden onderbroken.

```
NEW
10 REM DIT PROGRAMMA BLIJFT UW NAAM
20 REM CONTINU AFDRUKKEN.
30 INPUT "HOE HEET U ";NAAM$
40 PRINT NAAM$;"-";
50 GOTO 40
RUN
HOE HEET U ? KAREL
KAREL-KAREL-KAREL-KA...
REL-KAREL-KAREL-KA...
...
```

– Het gehele beeldscherm wordt vol gezet, onderbreken met CONTROL-STOP.

Merk op dat GOTO standaard onder funktie-toets 3 aanwezig is.

SLEUTELWOORD

IF-THEN/GOTO-ELSE

soort . . . . **KOMMANDO**

schrijfwijze

```
      THEN <DIREKTE OPDRACHT>
IF<N> GOTO
      GO TO <REGELNUMMER>
```

betekenis

Voor goed begrip van IF...THEN/GOTO...ELSE dienen de hoofdstukken 5, 6 en 7 terdege te zijn doorgenomen.

Deze kombinatie van sleutelwoorden biedt de mogelijkheid om numerieke of alfanumerieke relaties te onderzoeken en daarop beslissingen te nemen. Alleen indien de tussen IF en THEN/GOTO opgenomen numerieke uitdrukking WAAR is (of ongelijk aan nul is) worden de kommando's na THEN uitgevoerd of wordt met het regelnummer achter GOTO vervolgd.

Voorbeeld:

```
NEW
10 INPUT "HOEVEEL IS 2 MAAL 5 ";UITKOMST
20 IF UITKOMST=10 GOTO 100
30 IF UITKOMST<10 THEN PRINT "TE LAAG":
GOTO 10
40 PRINT "TE HOOG":GOTO 10
100 PRINT "HOERA,GOED"
110 STOP
RUN
HOEVEEL IS 2 MAAL 5 ? 9
TE LAAG
HOEVEEL IS 2 MAAL 5 ? 112.5
TE HOOG
HOEVEEL IS 2 MAAL 5 ? 10
HOERA,GOED
```

```
Break in 110
Ok
```

Ook ingewikkelder zaken kunnen worden afgevraagd, bijvoorbeeld met behulp van de logische bewerkingen:

```
NEW
10 INPUT "DE LUCHT IS ";KLEUR$
20 INPUT "EN HET GRAS RUIKT ";GEUR$
30 IF KLEUR$="BLAUW" AND GEUR$="FRIS"
THEN PRINT "GOEDZO":STOP
40 PRINT "NOG EEN KEER PROBEREN":GOTO 10
RUN
DE LUCHT IS ? GEEL
EN HET GRAS RUIKT ? FRIS
NOG EEN KEER PROBEREN
DE LUCHT IS ? BLAUW
EN HET GRAS RUIKT ? FRIS
GOEDZO
Break in 30
Ok
```

Met de ELSE-optie kan worden gespecificeerd welke akties dienen te worden ondernomen indien de uitdrukking tussen IF en THEN/GOTO NIET WAAR (gelijk aan 0) is. Bijvoorbeeld:

```
NEW
10 INPUT "HOE HEET U ";NAAM$
20 IF LEN(NAAM$)=5 THEN PRINT "UW NAAM
IS PRECIES 5 LETTERS LANG" ELSE PRINT "
HALLO ";NAAM$;" HOE GAAT HET ERMEE ?":G
OTO 10
RUN
HOE HEET U ? JAN
HALLO JAN HOE GAAT HET ERMEE
HOE HEET U ? FRITS
UW NAAM IS PRECIES 5 LETTERS LANG
Ok
```

Merk op dat regel 30 en 40 uit het voorlaatste voorbeeld kunnen worden vervangen door één regel:

```
30 IF KLEUR$="BLAUW" AND GEUR$="FRIS"
THEN PRINT "GOEDZO":STOP ELSE PRINT
"NOG EEN KEER PROBEREN":GOTO 10
```

SLEUTELWOORD INKEY$

---

soort . . . . **SYSTEEMVARIABELE**

schrijfwijze

```
INKEY$
```

betekenis

Deze funktie geeft als resultaat het karakter van de laatst ingegeven toets. Indien niets werd ingegeven, geeft INKEY$ een lege waarde als resultaat. Bijvoorbeeld:

```
NEW
10 LET A$=INKEY$:IF A$="" THEN 10
20 PRINT "U GAF ";A$;" IN."
30 GOTO 10
RUN
```

(De computer wacht. Zodra u een toets indrukt, meldt hij dat waarna hij weer wacht. Programma onderbreken met CONTROL-STOP).

N.B.: Zodra INKEY$ één maal is gebruikt, is het betreffende karakter verdwenen. Daarom moet INKEY$ steeds eerder in een alfanumerieke variabele worden overgebracht (zie regel 10) en daarna pas worden afgevraagd.

Op de CONTROL-BREAK-intoetsing na, worden alle toetsaanslagen bij het gebruik van INKEY$ geregistreerd. Voor controletoetsen wordt

ook een karakter gegeven. Om dit karakter te kunnen bestuderen, kunnen we de ASCII-waarde van dit karakter als volgt afvragen:

```
NEW
10 LET A$=INKEY$:IF K$="" THEN 10
20 PRINT "ASCII-WAARDE:";ASC(A$)
30 GOTO 10
RUN
ASCII-WAARDE: 13  (de RETURN-toets werd ingedrukt)
ASCII-WAARDE: 27  (de ESC-toets werd ingedrukt)
ASCII-WAARDE: 24  (de SELECT-toets werd ingedrukt)
```

(etcetera; probeer hoofdletters, kleine letters, controletoetsen en ook eens de funktietoetsen. Controleer het een en ander met hoofdstuk 10. en 18.)

SLEUTELWOORD

INPUT

---

soort . . . . **KOMMANDO**

schrijfwijze

```
INPUT["<SATELLIETTEKST>";]<VARIABELE>
  {,<VARIABELE>}
```

betekenis

Met het INPUT-kommando kunnen gegevens via het toetsenbord worden ingevoerd. Deze worden dan in een variabele opgeslagen. Voorbeeld:

```
NEW
10 INPUT A
20 PRINT A
30 STOP
RUN
? 12.34
```

– De computer geeft een vraagteken. Een

```
 12.34
Break in 30
Ok
```

waarde dient te worden ingetoetst.

De in te voeren gegevens kunnen numeriek of alfanumeriek zijn. Bij invoer van alfanumerieke gegevens dient het aanhalingsteken als eerste karakter te worden vermeden omdat MSX-basic hierop niet correct reageert. Voorbeeld:

```
NEW
10 INPUT A$
20 PRINT "HALLO ";A$
RUN
? FRANS
HALLO FRANS
Ok
```

Indien meerdere gegevens dienen te worden ingevoerd, kan dat in één INPUT-kommando gebeuren. De variabelen achter INPUT dienen dan met een komma van elkaar te zijn gescheiden. Voorbeeld:

```
NEW
10 INPUT A,B,C
20 PRINT A,B,C
30 GOTO 10
RUN
? 1,2,3.5
 1  2  3.5
? 22
?? 33
?? 44
 22  33  44
? 1,2,3,4
?Extra ignored
 1  2  3
?
```

– De gegevens mogen in één keer achter elkaar worden ingegeven, gescheiden door een komma.
– De gegevens mogen ook elk met een enter worden afgesloten. Met een dubbel vraagteken vraagt de computer dan een volgende ingave totdat alle variabelen zijn gevuld.
– Indien te veel waarden worden ingevoerd geeft de computer eerst een melding dat het teveel aan ingevoerde gegevens wordt overgeslagen.

De komma is het scheidingsteken bij INPUT. Daarom kan een komma

als *teken* via de input nooit in een stringvariabele worden overgeplaatst. Een INPUT-kommando mag worden voorzien van een begeleidende tekst. Bijvoorbeeld:

```
?
 12.5          – Alleen een RETURN wordt ingegeven.
               – En de oude waarde bleef behouden.
?
```

SLEUTELWOORD

INT

---

soort . . . . **N-FUNKTIE**

schrijfwijze

```
INT(<N>)
```

betekenis

Het resultaat van deze funktie is de grootste gehele waarde, kleiner dan of gelijk aan de waarde van de tussen haakjes vermelde numerieke uitdrukking. Om het verschil met FIX aan te geven, zijn beide in het voorbeeld verwerkt:

```
NEW
10 LET A=12.5:LET B=-12.5
20 PRINT INT(A);FIX(A)
30 PRINT INT(B);FIX(B)
RUN
 12  12
-13 -12
Ok
```

SLEUTELWOORD **KEY**

soort .... **KOMMANDO**

schrijfwijze

```
KEY LIST
    <FUNKTIETOETSNUMMER>,<UIT TE VOEREN FUNKTIE>\KEY(<...>
    (<FUNKTIETOETSNUMMER>) ON
                           OFF
```

betekenis

Het KEY-sleutelwoord heeft verschillende betekenissen. Hieronder volgt een beschrijving per betekenis:

Eerste betekenis: het uitlijsten van de inhoud van de funktietoetsen

Door ingave via KEY LIST verschijnen onder elkaar op het beeldscherm de funkties die onder de funktietoetsen liggen opgeslagen. Eventuele controlekarakters (zoals de in CHR$(13) gekodeerde RETURN toets, zie de tweede betekenis) worden als spaties afgedrukt.
Wanneer we KEY LIST op een zojuist aangeschakelde MSX-computer gebruiken, krijgen we de volgende lijst:

```
KEY LIST
color
auto
goto
list
run
color 15,4,4
cload"
cont
list.
 run
Ok
```

Wanneer we de funktietoetsen zelf indrukken, verschijnen de afgebeelde funkties daadwerkelijk. Zo heeft funktietoets F10 tot gevolg dat het beeldscherm eerst wordt schoongemaakt (controlekarakter vóór run; zie de afgebeelde spatie) waarna het programma wordt opgestart.

Tweede betekenis: het veranderen van de funktie van een funktietoets

Via het KEY sleutelwoord, gevolgd door een numerieke uitdrukking die niet met een haakje begint, gevolgd door een komma en een alfanumerieke uitdrukking, kan de funktie van een funktietoets worden veranderd. Bijvoorbeeld:

```
KEY 2,"TRON"
Ok
```

Na het intikken van dit kommando heeft funktietoets 2 de funktie TRON verkregen. Wanneer we hierna deze funktietoets indrukken, verschijnt de tekst TRON. Er dient nog wel een RETURN te worden ingegeven om het kommando te bevestigen.

De RETURN-toets correspondeert met een ASCII-kode 13. Deze toets kan als karakter dus worden meegegeven in een funktie. Bijvoorbeeld:

```
KEY 2,"TRON"+CHR$(13)
Ok
```

Wanneer we nu funktietoets 2 indrukken, verschijnt wederom het TRON-kommando maar wordt daarbij ook meteen een return gegeven. Direkt na de funktietoets verschijnt de Ok-melding; het TRON-kommando is uitgevoerd.

Wanneer we een funktie willen creëren die een TRON en vervolgens een RUN activeert, kunnen we dat bijvoorbeeld als volgt bewerkstelligen:

```
KEY 2,"TRON"+CHR$(13)+"RUN"+CHR$(13)
Ok
```

Het ligt voor de hand dat we funktietoets 1 als volgt inrichten:

```
KEY 1,"TROFF"+CHR$(13)+"RUN"+CHR$(13)
Ok
```

Indien we funktietoets 3 de funktie 'maak het beeld schoon en doe een LIST' willen geven, kan dat als volgt:

```
KEY 3,"CLS"+CHR$(13)+"LIST"+CHR$(13)
Ok
```

of

```
KEY 3,CHR$(12)+"LIST"+CHR$(13)
Ok
```

ASCII-kode 12 heeft tot effect dat het beeld wordt schoongemaakt.

De numerieke uitdrukking voor de komma geeft het funktietoetsnummer aan en dient een gehele, positieve waarde te bevatten, kleiner dan 11. Indien de numerieke uitdrukking een gebroken waarde bevat, wordt de decimale fractie verwaarloosd.

Derde betekenis; het aan- of uitzetten van de funktietoetsregel

De MSX-computer heeft standaard de beeldschermregel (in de alfanumerieke toestand) gereserveerd voor het presenteren van (een gedeelte van) de funkties van de funktietoetsen. Zolang deze funkties daar worden geprojekteerd, kan die onderste regel niet worden gebruikt door het programma.

Door eenvoudigweg het kommando

```
KEY OFF
```

in te toetsen, kunnen we de onderste regel leeg maken en vrij maken voor gebruik in een programma. Door het

```
KEY ON
```

kommando halen we de funktietoetsregel weer te voorschijn en wordt deze regel weer verboden terrein voor het programma.

SLEUTELWOORD — LEFT$

soort . . . . **A-FUNKTIE**

schrijfwijze

```
LEFT$(<BASISSTRING>,<AANTAL TEKENS>)
```

betekenis

Deze funktie geeft van een alfanumerieke uitdrukking het linkergedeelte als resultaat. Hoe groot dit linkergedeelte is, wordt bepaald door het opgegeven aantal tekens. Bijvoorbeeld:

```
PRINT LEFT$("MSX-BASIC",4)
MSX-
Ok
```

Het aantal tekens moet groter dan of gelijk aan nul zijn en mag de waarde 255 niet overschrijden. Een eventuele decimale fractie wordt verwaarloosd.

SLEUTELWOORD — LEN

soort . . . . **N-FUNKTIE**

schrijfwijze

```
LEN(<TE METEN STRING>)
```

betekenis

Deze funktie geeft als resultaat de lengte van de te meten string. Spaties en niet afdrukbare karakters die in de te meten string (een alfanumerieke uitdrukking) voorkomen, worden meegeteld.

Voorbeeld:

```
NEW
10 INPUT "HOE HEET U ";NAAM$
20 PRINT "UW NAAM IS";LEN(NAAM$);"LETTER
S LANG"
RUN
HOE HEET U ? FERDINAND
UW NAAM IS 9 LETTERS LANG
Ok
```

SLEUTELWOORD LET

soort . . . . **KOMMANDO**

schrijfwijze

```
[LET]<NUMERIEKE VARIABELE>=<N>¦[LET]<AL-
  FANUMERIEKE VARIABELE>=<A>
```

betekenis

Met het LET-kommando kan aan een variabele een bepaalde waarde worden toegekend. Indien een variabele nog niet eerder werd genoemd, wordt deze automatisch gecreëerd.

Bijvoorbeeld:

```
NEW
10 LET A=4
20 LET A$="TEST"
30 PRINT A;A$
RUN
 4 TEST
Ok
```

Het sleutelwoord LET mag volledig worden weggelaten. De volgende twee programmaregels zijn dan ook volledig identiek:

```
20 LET A$="TEST"
20 A$="TEST"
```

Indien een variabele zowel rechts als links van het eerste gelijk-teken voorkomt, dan wordt eerst de uitdrukking rechts van het eerste gelijk-teken uitgerekend en wordt daarna pas het eindresultaat aan de betreffende variabele toegekend. Voorbeeld:

```
NEW
10 INPUT "WAARDE ";WAARDE
20 LET WAARDE=WAARDE+WAARDE+WAARDE
30 PRINT "DRIE MAAL DEZE WAARDE IS";WAARDE
40 END
RUN
WAARDE 12.5
DRIE MAAL DEZE WAARDE IS 37.5
Ok
```

SLEUTELWOORD

LINE

---

soort .... **KOMMANDO**

schrijfwijze

```
LINE[@][<LOCATIE>]-<LOCATIE>[,[<KLEUR>][,
  B|,BF]\LINE<...>,
<LOCATIE>::=[STEP(<HORIZONTAAL>,<VERTI-
  KAAL>)]
```

betekenis

Voor goed begrip van dit kommando is het noodzakelijk dat eerst de PSET-behandeling volledig is doorgenomen.

Met dit kommando kunnen volledige lijnen of rechthoeken op het beeldscherm worden getekend. Een lijn wordt getrokken vanuit de eerste locatie naar de tweede in het LINE-kommando aangegeven

locatie. Indien de eerste locatie niet wordt opgenomen, dan wordt het laatste getekende punt als beginlocatie genomen. Beide locaties mogen zich buiten het beeldscherm bevinden. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 LINE (10,10)-(100,100)
30 GOTO 30
RUN
```

(een lijn wordt getrokken)

```
20 LINE (10,10)-STEP(90,90)
RUN
```

(alleen regel 20 werd veranderd. Dezelfde lijn wordt getrokken)

```
15 PSET (-10,-10)
20 LINE -STEP(100,100)
RUN
```

Regel 15 werd toegevoegd en regel 20 werd veranderd. Een lijn wordt getrokken vanaf het denkbeeldige punt (–10,–10) naar (90,90).

```
15
20 LINE@(-100,-100)-(500,500)
RUN
```

Regel 15 werd weer verwijderd en regel 20 werd veranderd. Een lijn wordt tussen de twee denkbeeldige punten getrokken.

De toevoeging van het karakter @ is toegestaan. Deze toevoeging dient om LINE te onderscheiden van LINE INPUT en heeft verder geen funktie. We zullen deze toevoeging in alle voorbeelden verder weglaten.

Indien bij het LINE-kommando geen kleur wordt gespecificeerd, wordt de voorgrondkleur die op dat moment actief is, genomen als tekenkleur. Uiteraard kan een andere voorgrondkleur worden geactiveerd. Bijvoorbeeld:

```
NEW
10 COLOR 15,4,4
20 SCREEN 2
30 LINE (1,1)-(233,120),6
40 GOTO 40
RUN
```

Een donkerrode streep wordt op een donkerblauwe achtergrond getekend. Merk op dat het COLOR-kommando VOOR het SCREEN-kommando werd gegeven om de juiste achtergrondkleur en randkleur te realiseren.

Indien de letter B (van BLOCK) wordt opgenomen, wordt er plotseling geen lijn meer getrokken maar wordt de rechthoek getekend waarvan de twee aangegeven punten respectievelijk de linker bovenhoek en de rechter onderhoek vormen. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 LINE (20,20)-(200,200),12,B
30 GOTO 30
RUN
```

Een donkergroene rechthoek wordt getekend.

```
20 LINE (20,20)-(200,200),,B
RUN
```

Alleen programmaregel 20 werd gewijzigd. De rechthoek wordt nu in de geactiveerde voorgrondkleur getekend.

Indien een rechthoek gedeeltelijk of geheel buiten het beeldschermbereik valt, worden de buiten het beeld vallende lijnen van de rechthoek toch getekend. Dit gebeurt aan de corresponderende uiterste rand van het beeldscherm. Dit is onder meer in het bovenstaande voorbeeld het geval.

Indien de letters BF (block filled) worden opgenomen, wordt de rechthoek ook nog ingekleurd. Bijvoorbeeld:

```
20 LINE (20,20)-(200,200),12,BF
```

Alleen regel 20 werd vervangen. De rechthoek uit het voorlaatste voorbeeld wordt nu ook donkergroen ingekleurd.

SLEUTELWOORD # LIST

soort . . . . **KOMMANDO**

schrijfwijze

```
LIST[&lt;REGELNUMMER>][-[&lt;REGELNUMMER>]]
```

betekenis

Met dit kommando kunt u op het beeldscherm:

- Het volledige programma laten lijsten. Geef in dat geval slechts LIST in.
- Het eerste gedeelte van uw programma laten lijsten. Geef in dat geval LIST in, gevolgd door een minteken en het laatste te lijsten regelnummer.
- Het laatste gedeelte van uw programma laten lijsten. Geeft u in dat geval LIST in, gevolgd door het eerste te lijsten regelnummer en een minteken.
- Een middengedeelte van uw programma laten lijsten. Geeft u in dat geval LIST in, gevolgd door het eerste te lijsten regelnummer, een minteken en het tweede te lijsten regelnummer
- één enkel regelnummer laten lijsten. Geef in dat geval LIST in, gevolgd door het betreffende regelnummer of een punt voor het laatste behandelde regelnummer.

Voorbeelden:

```
LIST
```

lijst het gehele programma

```
LIST -100
```

lijst alle regels tot en met regel 100

```
LIST 100-
```

lijst alle regels vanaf regel 100

```
LIST 100-200
```

lijst alle regels vanaf 100 tot en met 200

```
LIST 100
```

lijst alleen regel 100

```
LIST
```

lijst het laatst behandelde regelnummer

De aangegeven regelnummers behoeven niet noodzakelijkerwijs aanwezig te zijn.

LIST zit in twee verschillende uitvoeringen standaard onder de funktietoetsen 4 en 9.

SLEUTELWOORD

LOCATE

---

soort . . . . **KOMMANDO**

schrijfwijze

```
LOCATE[<TEKENPOSITIE>][,[<REGELPOSITIE>]
 [,<CURSOR>]]\LOCATE[<...>,]
```

betekenis

Met deze funktie kan de cursor gepositioneerd worden op een bepaalde plaats op het beeldscherm. Voorbeeld:

```
NEW
10 FOR X=0 TO 24:FOR Y=0 TO 20
20 LOCATE X,Y:PRINT "*"
30 NEXT Y,X:STOP
RUN
```

(In de linkerbovenhoek van het beeldscherm worden van boven naar beneden sterren afgedrukt.)
Merk op dat positie 0,0 wordt gevormd door de linkerbovenhoek van het scherm.

Eén van de twee beeldscherm-coördinaten mag worden weggelaten. Bijvoorbeeld:

```
NEW
10 LOCATE 20:PRINT "HALLO";
20 LOCATE ,0:PRINT "HIER BEN IK"
RUN
```

(De tekst HALLO wordt 20 posities uit de kantlijn afgedrukt.) Op de eerste regel van het beeldscherm wordt, vier posities verder uit de kantlijn, de tekst HIER BEN IK afgedrukt.

Met LOCATE kan ook de cursor zichtbaar of onzichtbaar worden gemaakt tijdens het projekteren. Het derde element achter LOCATE dient dan een 0 (onzichtbaar) of een positieve waarde, ongelijk aan 0 maar kleiner dan 256 (zichtbaar) te bevatten. Bij het opstarten is de cursor standaard onzichtbaar.

Indien alleen de cursor dient te worden veranderd, dient een LOCATE „0 (onzichtbaar) of bijvoorbeeld een LOCATE „1 (zichtbaar) te worden gebruikt.

De aangestuurde cursorposities mogen de beeldschermgrenzen natuurlijk niet overschrijden. De betreffende beeldschermmaten worden bij SCREEN behandeld.

Nog wat voorbeelden:

`LOCATE 10,20,0`	de cursor wordt op positie 10 (horizontaal), 20 (vertikaal) geplaatst en de cursor wordt onzichtbaar
`LOCATE ,5,1`	de corsor verplaatst zich naar de vijfde regel en wordt zichtbaar
`LOCATE 11,,0`	de cursor verplaatst zich naar de elfde positie horizontaal en wordt onzichtbaar.

Tijdens een INPUT of een LINE INPUT is de cursor altijd zichtbaar, hoe ook ingesteld. LOCATE heeft geen zin wanneer een grafisch scherm actief is. Zie hiervoor SCREEN.

SLEUTELWOORD LOG

---

soort .... **N-FUNKTIE**

schrijfwijze

```
LOG(<N>)
```

betekenis

Deze funktie geeft als resultaat de natuurlijke logarithme van de waarde van de tussen haakjes genoemde numerieke uitdrukking. Bijvoorbeeld:

```
NEW
10 INPUT "WAARDE ";A
20 PRINT "NAT. LOG.=";LOG(A)
30 GOTO 10
RUN
WAARDE ? 12
NAT. LOG.= 2.4849066497879
WAARDE ? 2.7182818285
NAT. LOG.= 1.00000000015
WAARDE ? 0
NAT. LOG.=
Illegal function call in 20
Ok
```

Uiteraard dient de tussen haakjes vermelde waarde positief te zijn.

SLEUTELWOORD

MID$

soort . . . . **A-FUNKTIE**

schrijfwijze

```
MID$(<BASISSTRING>,<EERSTE TEKEN>[,<AAN-
  TAL TEKENS>])
```

betekenis

Deze funktie geeft als resultaat een middengedeelte van een alfanumerieke uitdrukking. Welk middengedeelte wordt gegeven als resultaat, hangt af van het gespecificeerde eerste teken en het gespecificeerde aantal tekens. Bijvoorbeeld:

```
PRINT MID$("MSX-BASIC-COMPUTER",5,5)
BASIC
Ok
```

Het aantal tekens mag niet kleiner zijn dan nul en niet groter zijn dan 255. Het opgegeven eerste teken mag niet kleiner zijn dan 1 en niet groter dan 255. Eventuele decimale fracties worden verwaarloosd.

Indien het aantal tekens niet wordt opgegeven, dan wordt vanaf het eerste teken de rest van de waarde van de alfanumerieke uitdrukking als resultaat gegeven. Bijvoorbeeld:

```
NEW
10 LET A$="DIT IS EEN TEST"
20 LET B$=LEFT$(A$,3)
30 LET C$=MID$(A$,5,2)
40 LET D$=MID$(A$,8,3)
50 LET E$=MID$(A$,12)
60 PRINT C$;" ";B$;" ";D$;" ";E$;"?"
RUN
IS DIT EEN TEST?
Ok
```

SLEUTELWOORD **MOTOR**

soort .... **KOMMANDO**

schrijfwijze

MOTOR [ON / OFF]

betekenis

Alle cassetterecorderbesturende opdrachten in MSX kunnen de cassetterecordermotor ook aan- of uitschakelen. Hiertoe is een speciale REMOTE-aansluiting (afstandsbediening) voorgeschreven onder MSX. Bij een CSAVE-kommando en een CLOAD-kommando zien we, dat de cassetterecorder door de computer automatisch wordt ingeschakeld totdat de opdracht is voltooid. Na de opdracht wordt de cassetterecordermotor weer uitgeschakeld.

De cassetterecordermotor kan ook apart worden in- en uitgeschakeld zonder dat er direkt geladen of geschreven wordt. In dat geval dienen we het MOTOR-kommando te gebruiken. Wanneer we bijvoorbeeld de cassette in de recorder terug willen spoelen, kunnen we het afstandsbedieningsstekkertje uit de recorder halen en dan daadwerkelijk terugspoelen. Gemakkelijker is het om eerst

```
MOTOR ON
```

in te toetsen en dan terug te spoelen; de cassetterecorder is ingeschakeld. Om de cassetterecorder weer uit te schakelen, geven we in:

```
MOTOR OFF
```

In een programma kan het MOTOR-kommando worden gebruikt bij begeleide bediening van de cassetterecorder, bijvoorbeeld:

Het MOTOR-kommando kent drie verschijningen:

MOTOR ON de cassetterecorder wordt ingeschakeld
MOTOR OFF de cassetterecorder wordt uitgeschakeld
MOTOR de cassetterecorder wordt ingeschakeld indien deze uit-

geschakeld was en wordt uitgeschakeld indien deze ingeschakeld was.

In een programma kan het MOTOR-kommando worden gebruikt bij begeleide bediening van de cassetterecorder.

SLEUTELWOORD # NEW

---

soort . . . . **KOMMANDO**

schrijfwijze

```
NEW
```

betekenis

Het NEW-kommando heeft tot gevolg dat het programmageheugen geheel wordt schoongewist. Ook variabelen worden verwijderd en alle openstaande kanalen worden gesloten. Na een NEW komt de computer altijd met de Ok-melding, ook wanneer de NEW in een regel werd opgenomen.

Voorbeeld:

```
NEW
10 PRINT "HALLO ALLEMAAL"
30 NEW
RUN
HALLO ALLEMAAL
Ok
LIST
Ok
```

(programma is verdwenen)

SLEUTELWOORD

ON-GOSUB

soort . . . . **KOMMANDO**

schrijfwijze

ON<RANGWAARDE>GOSUB<REGELNUMMER>{,<REGEL-NUMMER>}

betekenis

Het is noodzakelijk om eerste de behandeling van ON...GOTO goed door te nemen alsmede die van het GOSUB-kommando.

Dit kommando werkt ongeveer hetzelfde als het ON...GOTO-kommando met dit verschil dat er subroutines worden aangesproken. Deze subroutines of onderprogramma's eindigen allemaal met het RETURN-kommando. Na het RETURN-kommando wordt verdergegaan met het kommando na de ON...GOSUB tenzij in het RETURN-kommando iets anders is aangegeven.

Bijvoorbeeld:

```
NEW
10 FOR I=1 TO 4
20 ON I GOSUB 100,200,,300
30 NEXT I:FOR I=1 TO 1E20:NEXT I
100 SCREEN 2:LINE (0,0)-(100,100):RETURN
200 CIRCLE (100,100),50:RETURN
300 LINE (10,10)-(60,40),,B:RETURN
RUN
```

Door elkaar worden een lijn (eerste subroutine), een cirkel (tweede subroutine) en een rechthoek (derde subroutine) getekend. Hierna staat de computer vast in regel 30.

SLEUTELWOORD # ON-GOTO

soort . . . . **KOMMANDO**

schrijfwijze

ON<RANGWAARDE> GOTO / GO TO <REGELNUMMER>
{,<REGELNUMMER>}

betekenis

Het is noodzakelijk dat eerste het GOTO-kommando is doorgenomen.

Met dit kommando is het mogelijk om afhankelijk van de waarde een numerieke uitdrukking naar diverse locaties in het programma te springen. Hiertoe worden de volgende acties bij een ON...GOTO ondernomen:

- allereerst wordt de numerieke uitdrukking uitgewerkt. Het resultaat wordt ontdaan van een eventuele decimale fractie.
- vervolgens wordt gecontroleerd of deze waarde groter is dan of gelijk is aan nul. Bij een negatieve waarde volgt een foutmelding.
- daarna wordt gecontroleerd of de berekende waarde niet groter is dan 255. Indien dit wel het geval is, volgt een foutmelding.
- de berekende waarde geeft aan, naar welk regelnummer moet worden gesprongen. Zo wordt bij de waarde 1 naar het eerste regelnummer gesprongen en bij de waarde 2 naar het tweede regelnummer enzovoorts. Indien geen regelnummer voor de betreffende waarde is gespecificeerd, wordt het programma vervolgd met het kommando na de ON...GOTO.

Bijvoorbeeld:

```
NEW
10 REM PROGRAMMAKEUZE
20 SCREEN 0:PRINT "PROGRAMMAKEUZE"
:PRINT:PRINT
30 PRINT "1...TEKEN EEN LIJN"
40 PRINT "2...TEKEN EEN CIRKEL"
```

```
50 PRINT "4...TEKEN EEN RECHTHOEK"
60 INPUT "KEUZE ";KEUZE
70 ON KEUZE GOTO 100,200,,300
80 PRINT "ALLEEN EEN 1,2 OF 4 INGEVEN":
GOTO 60
90 K$=INKEY$:IF K$="" THEN 90 ELSE 20
100 REM LIJN
110 SCREEN 2:LINE (0,0)-(200,100):
GOTO 90
200 REM CIRKEL
210 SCREEN 2:CIRCLE (100,100),50:
GOTO 90
300 REM RECHTHOEK
310 SCREEN 2:LINE (20,20)-(100,100),,B:
GOTO 90
RUN
```

Op het beeldscherm verschijnt een keuzemenu. Naar keuze kunnen een lijn, een cirkel of een rechthoek worden getekend. Nadat de tekening klaar is, behoeft slechts een toets te worden ingedrukt om weer het keuzemenu te verkrijgen.

De dubbele komma in regel 70 laat zien dat regelnummers kunnen worden overgeslagen. Pas bij de waarde 4 voor variabele KEUZE wordt naar regel 300 gesprongen.

SLEUTELWOORD PAINT

soort . . . . **KOMMANDO**

schrijfwijze

```
PAINT<LOCATIE>[,[<SCHILDERKLEUR>][,<RAND-
  KLEUR>]]\PAINT[<...>,]
<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTI-
  KAAL>)
```

betekenis

Met dit kommando kunnen binnen een tekening op scherm gehele vlakken worden ingekleurd. Door slechts een punt aan te geven, eventueel een schilderkleur te kiezen en eventueel een randkleur te kiezen, wordt het betreffende oppervlak ingekleurd.

Voor goed begrip van dit kommando is het noodzakelijk om de behandeling van PSET grondig door te nemen.

De wijze waarop PAINT werkt, kunnen we het beste vergelijken met de wijze waarop we zelf een tekening inkleuren.
Wanneer we een oppervlak gaan inkleuren is het noodzakelijk om eerst het juiste kleurpotlood te kiezen, om de juiste kleur te bepalen. Vervolgens dienen we te bepalen waar we het kleurpotlood op papier gaan neerzetten. Als laatste moeten we beslissen tot aan welke lijnen we gaan inkleuren.
Met het PAINT-kommando geven we het punt waar het kleurpotlood moet worden neergezet aan met de betreffende locatie. Met de schilderkleurkodering bepalen we de kleur. Met de randkleurkodering bepalen we de kleur van de lijnen waarbinnen de computer met kleuren dient te blijven.

Indien de schilderkleur wordt weggelaten, wordt aangenomen dat de schilderkleur gelijk is aan de op dat moment actieve voorgrondkleur. Indien de randkleur wordt weggelaten, wordt aangenomen dat ook deze kleur gelijk is aan de op dat moment actieve voorgrondkleur.

Indien de hoge resolutie is geactiveerd (SCREEN 2) dan is het overbodig om een randkleur te specificeren; de schilderkleur wordt altijd als randkleur genomen.

Enige voorbeelden:

```
NEW
10 SCREEN 2
20 CIRCLE (111,111),75
30 CIRCLE (200,200),100
40 PAINT (160,160)
50 GOTO 50
RUN
```

Twee cirkels worden getekend. Het snij-oppervlak wordt ingekleurd.

```
10 SCREEN 3
40 PAINT (160,160),12
RUN
```

Alleen regels 10 en 40 werden veranderd. In lage resolutie worden de twee cirkels nu ingekleurd. Het snij-oppervlak wordt donkergroen gekleurd.

```
20 CIRCLE (111,111),75,12
30 CIRCLE (200,200),100,12
40 PAINT (160,160),4,12
RUN
```

Programmaregels 20, 30 en 40 werden vervangen. Twee groene cirkels worden getekend. Het snij-oppervlak wordt donkerblauw ingekleurd.

```
40 PAINT (160,160),,12
RUN
```

Alleen regel 40 werd vervangen. Het snij-oppervlak wordt met de op dat moment actieve voorgrondkleur ingekleurd.

Merk op dat het met PAINT voldoende is om ergens binnen het in te kleuren vlak een punt te prikken.

```
20 CIRCLE (111,111),75,0
30 CIRCLE (200,200),100,0
40 PAINT (160,160),12,0
RUN
```

De regels 20, 30 en 40 werden vervangen. De cirkels werden in de transparantkleur (onzichtbaar) getekend. Daarna werd het snijvlak ingekleurd. Alleen het ingekleurde snijvlak is zichtbaar. Dit laatste voorbeeld is uitstekend om te zien hoe de transparantkleur (kleurkode 0) funktioneel kan worden gebruikt.

SLEUTELWOORD # PLAY

soort .... **KOMMANDO**

schrijfwijze

```
PLAY<EERSTE STEM>[,<TWEEDE STEM>[,
  <DERDE STEM>]]
```

betekenis

Met het PLAY-kommando kunnen we op eenvoudige wijze een maximaal driestemmig stuk muziek vanaf een muziekmanuscript programmeren in de computer. Afhankelijk van het aantal stemmen dienen hiertoe achter het PLAY-kommando één, twee of drie alfanumerieke uitdrukkingen te zijn opgenomen.
Een geluids*effect*, bijvoorbeeld als begeleiding bij spelletjes, kan beter met het SOUND-kommando worden geprogrammeerd; zie aldaar.
De alfanumerieke uitdrukkingen dienen te zijn opgebouwd zoals de MML regels dat voorschrijven. MML (Music Macro Language) is te beschouwen als een aparte taal binnen MSX-basic, ontworpen door Digital Research, één van de grotere wereld-software producenten.

MML gaat uit van de volgende toonhoogte-symbolen:

C,D,E,F,G,A,B	deze letters stellen de namen van de noten voor van het te programmeren stuk muziek.
+, –, #	met een plus- of een min-teken achter de naam van de noot geeft men aan of deze halve toon moet worden verhoogd of verlaagd. Een plusteken komt dus overeen met een kruis en een minteken komt overeen met een mol. Dubbelmollen en dubbelkruizen zijn niet toegestaan. Het plus-teken mag door het echte kruis-teken worden vervangen.

Voorbeeld: de chromatische toonladder:

```
PLAY "CC+DD+EFF+GG+AA+B"
Ok
```

Achtereenvolgens worden de C, de Cis, de D, de Dis, de E, de F, de Fis, de G, de Gis, de A, de Ais en de B gespeeld door de computer.

Technisch heeft de volgende ingave exact hetzelfde effect:

```
PLAY "CD-DE-EFG-GA-AB-B"
Ok
```

Exact dezelfde klanken, voorstellende de C, de Des, de D, de Es, de E, de F, de Ges, de G, de As, de A, de Bes en de B, worden gespeeld.

Een zéér eenvoudig stukje driestemmige muziek:

```
NEW
10 PLAY "CDEFGABC","EFGABCDE","GABCDEFG"
RUN
```

Een zevental verschillende drieklanken wordt door de computer geproduceerd.

Door een getal (een integere konstante groter dan 0 en kleiner dan 65) achter de betreffende noot te plaatsen, bepaalt men de lengte van de noot. Een 1 staat voor een gehele noot, een 2 staat voor een halve noot, een 3 staat voor een derde noot etcetera.
In het voorbeeld worden enkele verschillende noten gespeeld; een hele, een halve, een kwart, een achtste, een zestiende en een tweeëndertigste noot:

```
PLAY "C1D2E4F8G16A32"
Ok
```

Als standaard nootlengte wordt een kwartnoot aangenomen. Deze standaardlengte kan worden gewijzigd door het L-kommando. Met een L, gevolgd door een nootlengte (1 ... 64) wordt de standaard nootlengte voor de betreffende stem veranderd. Indien alleen de letter L wordt gebruikt, wordt een L4 aangenomen.

Het volgende voorbeeld resulteert in een snelle toonladder; de standaard nootlengte werd op een zestiende noot gezet:

```
PLAY "L16CDEFCAB"
Ok
```

Een rust wordt met het symbool R aangegeven, gevolgd door de duur van de rust (1 ... 64). Indien geen duur wordt opgegeven, wordt een

kwartrust uitgevoerd. Voorbeeld: altijd is kortjakje ziek, eerste regel, tweestemmig:

```
PLAY "M6000S1L4CCGGAAGRL8FFFFL4
EEDDCR","M6000S1L4RREEFFERL8DD
DDL4CCGGER"
Ok
```

Op de M6000S1 kodering die in beide stemmen werd gebruikt, gaan we later in. Voorlopig is het van belang om slechts te weten, dat deze toevoeging ervoor zorgt dat de tonen afzonderlijk hoorbaar zijn.

Met het O-kommando (pas op: de letter O en niet het cijfer 0!) kunnen we het oktaaf sturen. Een oktaaf loopt altijd van C tot B. Indien geen oktaaf wordt gespecificeerd, wordt een vierde oktaaf aangenoment; het oktaaf waarin de centrale C zich als basis bevindt. Het volgende voorbeeld geeft de grote terts toonladder van C, klimmend over drie oktaven:

```
PLAY "O3CDEFGABO4CDEFGABO5CDEFGAB"
Ok
```

Achter het O-kommando dient dus een cijfer (1 ... 8) te worden opgenomen dat aangeeft in welk oktaaf de betreffende noot dient te worden gespeeld. Het volgende voorbeeld laat de laagst en hoogst mogelijke toon binnen deze notatie horen:
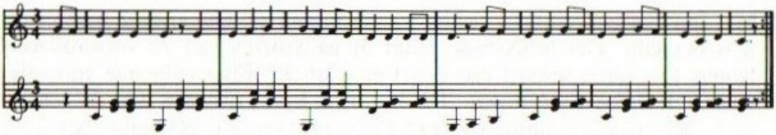
```
PLAY "O1CO8B"
Ok
```

Indien achter de letter O geen cijfer wordt opgenomen, wordt een O4 uitgevoerd.
Indien achter een noot een punt wordt geplaatst, wordt deze analoog aan de normale muzieknotatie met de halve lengte verlengd. Een tweede punt resulteert op dezelfde wijze in een tweede verlenging die de helft in lengte duurt van de eerste verlenging enzovoorts.

Voordat we op de wat meer technische kommando's van MML overgaan, volgt eerst een combinatievoorbeeld in de vorm van een eenvoudig driestemmig kinderliedje:

GOEDENAVOND SPEELMAN

MSX-ARRANGEMENT ♩=120



```
10 PLAY "V15T120","S1M9999T120",
"S1T120"
20 PLAY "L8O4GFL4EEEE.R8E","O4L4RCEEO3
GO4EE","O4L4RRGGRGG"
30 PLAY "EFGGFL8FEL4DDL8DD","CGGO3GO4G
GDFF","RO5CCRCCO4RGG"
40 PLAY "L4D.R8L8GFL4EEL8GFL4EEL8GF","
O3GABO4CEFCEF","RRRRO4GGRGG"
50 PLAY "L4ECDC.R8","CEFE.R8","RGGG.R8"
60 GOTO 20
RUN
```

Op de funktie van regel 10 gaan we later pas verder in.

Met het V-kommando kan per stem het volume worden bepaald. Achter de letter V dient dan een integere konstante te worden opgenomen, niet kleiner dan nul en niet groter dan 15. 15 geeft het hoogste volume, 0 geeft geen volume. Indien een V-kommando zonder achtervoegsel wordt gebruikt, wordt een V10 uitgevoerd. In het voorgaande voorbeeld zien we in regel 10 ondermeer de volume-instelling van de eerste stem. Omdat deze het beste gehoord dient te worden, wordt deze op het hardst (V15) gezet. Probeer het voorbeeld ook eens met V0...V14. Wanneer niet eerder een volume werd gespecificeerd, staat het volume standaard op V10.

Met het T-kommando kan het tempo van de muziek per stem worden bepaald. Achter de letter T dient dan een integere konstante, niet kleiner dan 32 en niet groter dan 255 te worden opgenomen. Dit getal stelt het aantal kwartnoten voor dat per minuut dient te worden gespeeld. Standaard staat het tempo ingesteld op 120 kwartnoten per seconde (T120). Het laatste voorbeeld kan door verandering van de T120 in regel 10 naar T255 (drie maal, één maal voor elke stem) tot

twee maal zo snel worden gespeeld door de computer. Indien alleen de letter T wordt gebruikt, dan wordt een T120 uitgevoerd.

Met het N-kommando kunnen toonhoogten op een andere wijze worden bepaald. Het MSX-basic staat de aansturing van 96 verschillende tonen toe, genummerd van 1 tot en met 96. Elke volgende toon ligt weer een halve toon hoger . De 36e toon komt overeen met de centrale C. Met het N-kommando kan een toon worden gespeeld niet door door de muzikale notatie te gebruiken maar door de juiste toon uit de rij van 96 te kiezen. Het nummer van deze toon dient dan achter de letter N te worden geplaatst.
Het volgende voorbeeld toont twee kommando's die precies dezelfde uitwerking hebben:

```
PLAY "O4L4CDEFGFEDC"
Ok
PLAY "NL436N38N40N41N43N41N40N38N36"
Ok
```

Een 0 achter de letter N resulteert in een rustpauze.

Met de letter X, gevolgd door een alfanumerieke variabele en een puntkomma, kan een alfanumerieke variabele worden ingelast. Hierdoor vormt ondermeer de maximale lengte van een alfanumerieke variabele (255 tekens) geen barrière meer. Een voorbeeld:

```
NEW
10 LET A$="CDEFGAB"
20 PLAY "O1XA$;O2XA$;O3XA$;O4XA$;
O5XA$;O6XA$;O7XA$;O8XA$;"
RUN
Ok
```

De grote terts toonladder wordt in alle mogelijke oktaven gespeeld.

Met het gelijkteken (=), gevolgd door een numerieke variabele en een puntkomma, kan de waarde van een numerieke variabele worden tussengevoegd. De variabele wordt pas tussengevoegd nadat een eventuele decimale fractie is verwaarloosd. Indien de waarde niet ligt binnen de toegestane waarden, volgt een foutmelding.
Het voorbeeld speelt alle mogelijke toonhoogten:

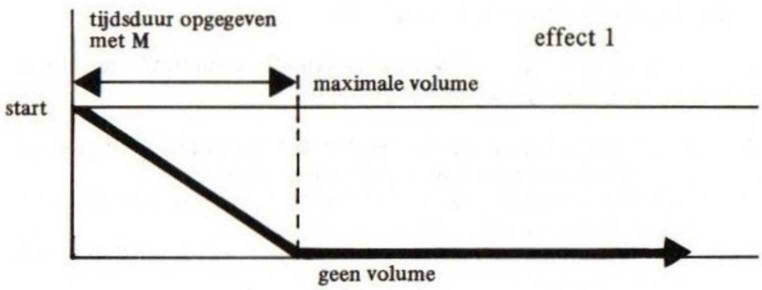
```
NEW
10 FOR I=1 TO 96
20 PLAY "N=I;"
30 NEXT
RUN
Ok
```

Met de letter S kan men één van de acht mogelijke volume-effecten selekteren. Het gebruik van de letter V naast de letter S heeft geen zin; een volume-effect heeft altijd een vast verloop.

Indien gebruik wordt gemaakt van een volume-effect, dan is het M-kommando daarbij onmisbaar. Met het M-kommando kan men de tijdseenheid specificeren waarbinnen een geluidseffect dient te zijn afgewikkeld. Hiertoe dient achter de letter M een gehele konstante te worden opgenomen, niet kleiner dan –65535 en niet groter dan 65535. Indien de tijdseenheid niet via een konstante maar via een numerieke variabele wordt gespecificeerd met gebruikmaking van het gelijkteken en de puntkomma, dan mag de waarde van de gebruikte variabele niet kleiner zijn dan –32768 en niet groter dan 32767.
Bij een negatieve waarde wordt voor uitvoering eerst de waarde 65536 opgeteld. De kenner herkent hierin de tweecomplementmethode.

De duur in seconden van deze tijdseenheid rekent men uit door de achter het kommando M gespecificeerde waarde te delen door de waarde 6965.

Een voorbeeld: volume-effect nummer 1 heeft tot gevolg dat het volume op de navolgende wijze verloopt:



Een toon begint dus op vol volume en sterft vervolgens weg. De tijd die verstrijkt totdat de toon volledig is weggestorven, wordt via het M-kommando bepaald. We laten nu een toon in één seconde wegsterven waardoor we een soort nagalm-effekt krijgen:

```
PLAY "L2M6965S1T32A"
Ok
```

Vervolgens laten we de toon in een tiende seconde wegsterven zodat een tokkel-effect ontstaat:

```
PLAY "L2M697S1A"
Ok
```

Volume-effect 8 is een continue herhaling van effect 1. Steeds wanneer de toon is uitgestorven, wordt deze weer op vol volume gebracht. We laten een toon tien maal per seconde uitsterven en weer op volume komen:

```
PLAY "L2M697S8A"
Ok
```

Een snel tokkeleffect ontstaat. We kunnen ook 6965 maal per seconde de toon laten uitsterven en weer op volume komen:

```
PLAY "L2M1S8A"
Ok
```

Door deze zeer snelle afwisseling ontstaat een schel geluidseffect.

Door veel met het M- en het S-kommando te combineren, kunnen allerlei geluidseffecten worden verkregen.

In de tabel op de volgende bladzijde staan alle geluidseffecten die in MSX-basic mogelijk zijn, bij elkaar genoemd.

Indien geen tijdsduur door middel van het M-kommando werd gespecificeerd, dan wordt een tijdsduur M255 aangenomen.
Indien een M-kommando zonder achtervoegsel wordt gebruikt, dan wordt de standaard waarde (255) aangenomen.
Indien een S-kommando zonder achtervoegsel wordt gebruikt, dan wordt een S0 uitgevoerd.
Een S-kommando kan worden opgeheven door gebruik van het V-

kommando. Voor alle drie de stemmen kan maar één effect (s) en één tijdsduur (M) worden gespecificeerd. Steeds worden de laatst gespecificeerde waarden als geldend aangenomen voor alle stemmen waarvoor een effect werd aangezet.

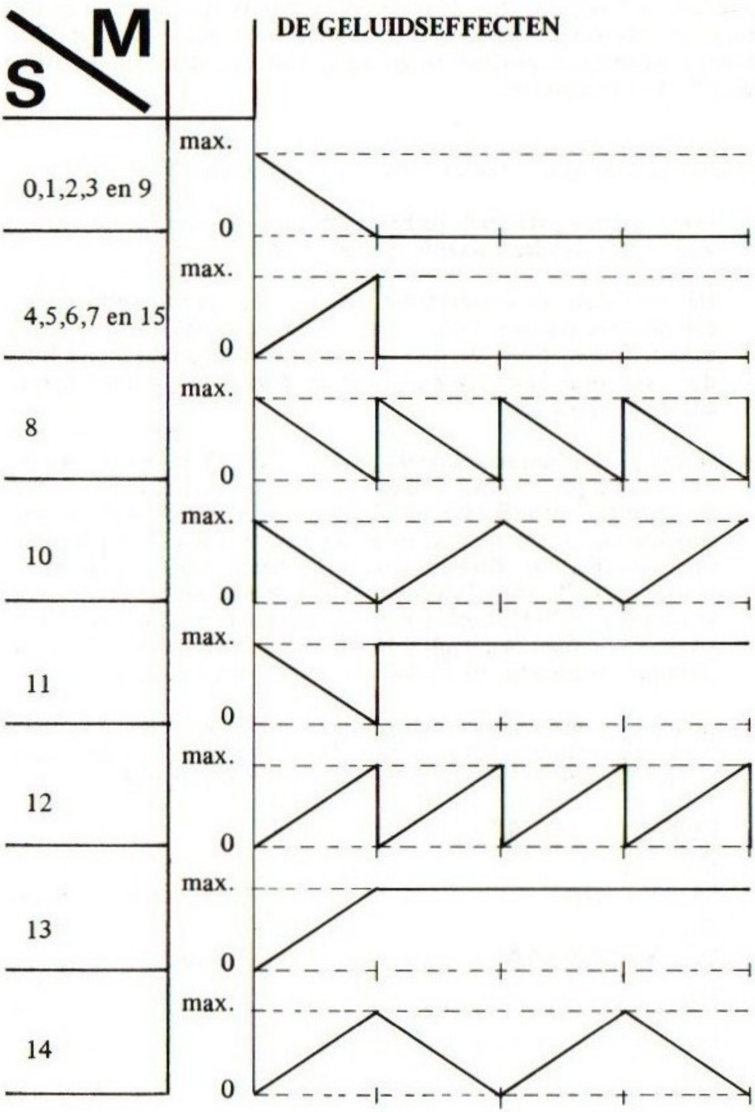
Enkele opmerkingen in verband met het PLAY-kommando:

- Door gebruik van het BEEP-kommando worden alle waarden weer op de standaardwaarde teruggebracht.
- Het maximale uitvoeringstempo is tien tonen per seconde. Indien een snellere passage wordt geprogrammeerd, blijft de uitvoering achter. Indien twee- of driestemmig wordt uitgevoerd, kan hierdoor een ongelijke uitvoering van de drie stemmen (een desynchronisatie) onstaat.
- Doordat besturingskommando's (zoals O, L, T en V) een kleine uitvoeringstijds inbeslag nemen, kan een desynchronisatie tussen de geprogrammeerde stemmen ontstaan bij meerstemmige programmering. Deze desynchronisatie kan worden opgelost door tussenvoeging van R64-kommando's (vierenzestigste rusten) op gehoor in de 'te snelle' stem. Ook verdient het aanbeveling om de stemmen per PLAY-kommando niet te lang te maken. Per PLAY-kommando vindt namelijk een algehele synchronisatie plaats; de stemmen worden bij elk PLAY-kommando altijd gelijk gestart.
- Indien geen effecten worden gebruikt, is de toonscheiding tussen twee gelijke tonen niet hoorbaar. Dit kan worden opgelost door een R64 tussen de twee noten in te lassen. Bijvoorbeeld:

```
PLAY "V10L4T120"
Ok
PLAY "AA"                 (een toon is hoorbaar)
Ok
PLAY "AR64A"              (twee tonen zijn hoorbaar)
Ok
```

  Pas met deze methode echter wel op voor desynchronisatie-effecten.

DE GELUIDSEFFECTEN



SLEUTELWOORD POS

soort .... **N-FUNKTIE**

schrijfwijze

```
POS(<U>)
<U>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Deze funktie geeft als resultaat de positie waarop de cursor zich op het moment in een regel bevindt. De uitdrukking tussen haakjes is een dummy-uitdrukking; in de praktijk kan het beste gewoon een nul worden gebruikt. Voorbeeld:

```
NEW
10 FOR I=0 TO 10 STEP 2
20 LOCATE I:PRINT POS(0)
30 NEXT I
RUN
 0
   2
     4
       6
         8
           10
Ok
```

SLEUTELWOORD PRINT

soort .... **KOMMANDO**

schrijfwijze

```
PRINT
  ?    [{,|;}<P>{,/;[<P>]}][
```

```
USING<USING STRING>;<U>{;<U>}
<KANAAL>::=<N>
<P>::=<U>|TAB(<POSITIE>)SPC(<AANTAL
  SPATIES>)
```

betekenis

Het PRINT-kommando is één van de meest belangrijke maar ook ingewikkelde kommando's. Het kommando is op veel zeer verschillende manieren te gebruiken.

eenvoudig gebruik

Met het PRINT-kommando kan op beeldscherm een regel worden opgeschoven. Voorbeeld:

```
PRINT:PRINT:PRINT:PRINT
```

(deze regels worden overgeslagen)

```
Ok
```

Met het PRINT-kommando kunnen konstanten of de inhoud van variabelen worden afgedrukt op het beeldscherm. Voorbeeld:

```
LET A=2.3:LET A$="TEST"
Ok
PRINT "DIT IS EEN ";A$;A
DIT IS EEN TEST 2.3
Ok
```

Indien de verschillende af te drukken uitdrukkingen van elkaar gescheiden zijn met een punt-komma, worden deze uitdrukkingen tegen elkaar aan afgedrukt. Merk op dat voor een positieve waarde altijd een spatie wordt afgedrukt. Deze spatie kan worden beschouwd als het teken (plus-teken) van deze waarde. Ná een numerieke waarde wordt ook altijd een extra spatie afgedrukt.
Indien de verschillende uitdrukkingen van elkaar gescheiden worden door een komma, wordt elke uitdrukking een kolom verder afgedrukt. Het beeldscherm is hiertoe ingedeeld in kolommen van veertien tekens. Bijvoorbeeld:

```
PRINT "EEN","TEST"
EEN           TEST
Ok
```

Een TAB-funktie biedt de mogelijkheid om zelf tabulaties (kolomindelingen) door te voeren. Tussen de haakjes van de TAB-funktie dient de positie te worden vermeld waarop het afdrukken op het beeldscherm dient te beginnen. Bijvoorbeeld:

```
NEW
10 PRINT "EEN";TAB(20);"TWEE"
20 PRINT "APPELS";TAB(20);"PEREN"
RUN
EEN                 TWEE
APPELS              PEREN
Ok
```

Een SPC-funktie biedt de mogelijkheid om tussen twee af te drukken uitdrukkingen een vast aantal spaties te plaatsen. Bijvoorbeeld:

```
NEW
10 PRINT "EEN";SPC(20);"TWEE"
20 PRINT "APPELS";SPC(20);"PEREN"
RUN
EEN                    TWEE
APPELS                    PEREN
Ok
```

Indien de tussen de haakjes van TAB of SPC vermelde waarde gebroken is, zal de hoogste gehele waarde kleiner dan deze gebroken waarde geldig zijn.

Wanneer een PRINT-kommando wordt afgesloten met een puntkomma, dan zal een volgende printopdracht achter het laatst afgedrukte gedeelte doorgaan en dus niet op een volgende regel beginnen. Bijvoorbeeld:

```
NEW
10 PRINT "HALLO ";
20 PRINT "ALLEMAAL"
```

```
RUN
HALLO ALLEMAAL
Ok
```

Wanneer een PRINT-kommando wordt afgesloten met een komma, dan zal een volgende printopdracht niet op de volgende regel maar in een volgende kolom doorgaan. Bijvoorbeeld:

```
NEW
10 PRINT "EEN",
20 PRINT "TWEE"
RUN
EEN            TWEE
Ok
```

N.B.: MSX-basic versie 1.0 maakt in het PRINT-bevel in sommige constructies een fout. Een enkele maal wordt een gegeven toch onbedoeld op een volgende regel gezet. De tweede kolom laat zich met een komma niet vinden; in plaats daarvan wordt op de volgende regel verder gegaan.

gevorderd gebruik: USING

Indien een PRINT-kommando wordt geconstrueerd met daarin het USING-kodewoord, dan kan direkt na dat USING-kodewoord een formaat (of print-masker) worden gespecificeerd. De af te drukken gegevens dienen dan naar dit formaat te worden gevormd.
De diverse karakters die in een formaatspecificatie (using-string) mogen worden gebruikt, worden hieronder per stuk in een voorbeeld behandeld.

Het "!"-teken

Met het "!"-teken kan worden aangegeven dat slechts één karakter dient te worden afgedrukt. Voorbeeld:

```
PRINT USING "!";"MSX"
M
Ok
```

Het "\"-teken

Met het "\"-teken kan worden aangegeven dat slechts een bepaald aantal karakters dient te worden afgedrukt. Hiertoe dienen in de using-string twee "\"-tekens te worden opgenomen met daartussenin het aantal af te drukken tekens minus 2 **aan spaties**. Voorbeeld:

```
NEW
10 LET A$="ABCDEFGHIJKLMNOP"
20 PRINT USING "\   \";A$
RUN
ABCDE
Ok
```

Zoals ook voor de nog te behandelen tekens geldt, mogen de diverse speciale tekens ook bij elkaar in één using-string worden gebruikt. Daarbij mogen ook niet-speciale karakters in een using-string worden gebruikt; deze worden dan gewoon afgedrukt. Bijvoorbeeld:

```
NEW
10 LET U$="\   \/!"
20 PRINT USING U$;"ABCDEFGH";"919191"
RUN
ABCDE/9
Ok
```

Het "&"-teken

Met het "&"-teken kan worden aangegeven dat een alfanumerieke uitdrukking onverkort dient te worden afgedrukt. Voorbeeld:

```
10 INPUT "WOORD ";A$
20 PRINT USING "DE EERSTE LETTER VAN & I
S !";A$;A$
RUN
WOORD ? KRAAI                    (dit wordt ingegeven)
DE EERSTE LETTER VAN KRAAI IS K
Ok
```

Het "#"-teken

Met het "#"-teken geeft men aan dat op die plaats een cijfer van de

uitkomst van een numerieke uitdrukking mag worden geplaatst. Met een veelheid van "#"-tekens kan men precies bepalen waar de uitkomst van een numerieke uitdrukking moet komen te staan en hoeveel posities deze mag hebben. Bijvoorbeeld:

```
10 FOR I=7 TO 12:PRINT USING "##";I
20 NEXT I
RUN
 7
 8
 9
10
11
12
Ok
```

Merk op dat in bovenstaand voorbeeld de getallen netjes **rechts** gericht onder elkaar staan. Zonder het gebruik van de USING-mogelijkheid zouden ze links gericht onder elkaar staan.
Indien de using-string te klein is, wordt het betreffende numerieke antwoord normaal afgedrukt maar met een voorlopend "%"-teken als waarschuwing. Bijvoorbeeld:

```
PRINT USING "### ##";100;100
100 %100                    (de tweede "using" is te klein)
Ok
```

Indien nodig, wordt het af te drukken getal door dit USING-gebruik afgerond. Bijvoorbeeld:

```
NEW
10 PRINT USING "##";1.4;1.5
 1 2
Ok
```

Uit het voorgaande voorbeeld blijkt ook dat wanneer de using-string is "opgebruikt", hij weer van voor af aan opnieuw wordt gebruikt. In dit geval wordt hierdoor twee maal hetzelfde patroon gebruikt voor het afdrukken van de getallen 1.4 en 1.5. Voor een negatieve waarde dient een extra positie te worden gecreëerd in verband met het te plaatsen "–"-teken. Bijvoorbeeld:

```
PRINT USING "##";10;-10
10%-10
```
(voor −10 is de using-string te klein).

De punt

Met de decimale punt kan het numerieke print-masker met decimale posities worden gebruikt. Ook bij dit gebruik wordt eventueel weer afgerond. Voorbeeld:

```
PRINT USING "###.##";11.2;12.125
 11.20 12.13
Ok
```

Merk op dat posities na de decimale punt eventueel netjes met nullen wordt aangevuld.

Het "+"-teken

Het "+-teken mag vóór of na een numeriek print-masker worden geplaatst. Dit heeft tot gevolg dat het teken van het af te drukken getal voor of na dit getal wordt vermeld. Bijvoorbeeld:

```
NEW
10 LET A=22.3:LET B=-111
20 PRINT USING "+####.##";A;B
30 PRINT USING "####.##+";A;B
RUN
 +22.30 -111.00
 22.30+ 111.00-
Ok
```

Het "–"-teken

Het "–"-teken heeft bijna dezelfde uitwerking als het "+"-teken. Het verschil is dat in plaats van een "+"-teken een spatie word afgedrukt in het uiteindelijke resultaat en dat het "–"-teken alleen rechts van een print-masker het gewenste effect heeft. Bijvoorbeeld:

```
PRINT USING "###.##-";1;-1
 1.00     1.00-
Ok
```

Het "**"-teken

Een dubbele ster, geplaatst voor een numeriek print-masker, heeft tot gevolg dat het print-masker met twee numerieke plaatsingsmogelijkheden voor cijfers wordt uitgebreid en dat daarbij alle spaties die door gebruik van dit print-masker voor het af te drukken getal zouden ontstaan, worden vervangen door sterren. Bijvoorbeeld:

```
PRINT USING "**##.##";2
***2.00
Ok
PRINT USING "** ";2;22
*2 22
Ok
```

Uit het laatste voorbeeld blijkt dat de dubbele ster ook alleen mag worden gebruikt.

Het "$$"-teken

Een dubbel dollarteken, geplaatst voor een numeriek print-masker, heeft tot gevolg dat er één numerieke plaatsmogelijkheid extra wordt gecreëerd voor een cijfer en dat vóór het af te drukken getal een dollarteken wordt geplaatst. Bijvoorbeeld:

```
PRINT USING "$$#.##";2.5
 $2.50
Ok
```

Ook het "$$"-teken mag eventueel alleenstaand worden gebruikt.

Het "**$"-teken

Het gebruik van twee sterren en één dollarteken heeft tot gevolg dat de "$$" en de "**" worden gecombineerd. Het resultaat is twee extra plaatsingsmogelijkheden voor cijfers, een dollarteken voor het af te drukken getal en een opvulling van de linker spaties met sterren. Bijvoorbeeld:

```
PRINT USING "**$#.##";2.5
**$2.50
Ok
```

Ook het "**$"-teken mag alleenstaand worden gebruikt.

De komma

Een komma, opgenomen in een numeriek print-masker, niet als eerste teken en links van de eventueel aanwezige decimale punt heeft tot gevolg dat de duizendtallen met een komma worden afgescheiden in het af te drukken getal. Eventueel mogen meerdere komma's op de beschreven wijze worden opgenomen; zij resulteren alle in extra plaatsingsmogelijkheden voor cijfers. Voorbeeld:

```
NEW
10 PRINT USING "####,.##";1000
20 PRINT USING "####.##,";1000
RUN
1,000.00
1000.00,
Ok
```

Merk op dat op programmaregel 20 de komma niet het beschreven effect heeft; de komma staat **na** de decimale punt.

Het "^^^^"-teken

Met het teken "^^^^" geeft men aan, dat een exponentiële vorm dient te worden gehanteerd bij het afdrukken. De "**", de "$$", de "**$" en het gebruik van de komma zijn in combinatie met "^^^^" wel toegestaan maar geven zinloze resultaten. Het gebruik van een "+" of een "–" **ACHTER** het numerieke print-masker is ook toegestaan maar geeft eveneens een zinloos resultaat.
"^^^^" dient direkt na het numerieke print-masker te worden opgenomen. Bijvoorbeeld:

```
PRINT USING "#.##^^^^";20000
0.20E+05
Ok
```

SLEUTELWOORD # PSET

soort . . . . **KOMMANDO**

schrijfwijze

```
PSET<LOCATIE>[,<KLEUR>]
<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTI-
  KAAL>)
```

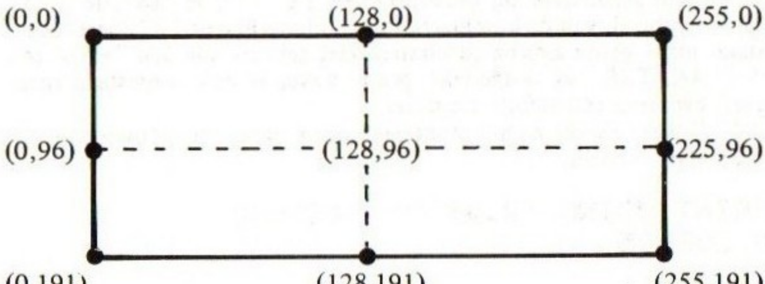
betekenis

Voor een goed begrip van dit kommando is het raadzaam, eerst het SCREEN- en het COLOR-kommando goed door te nemen.

Met dit kommando kan een puntje op een grafisch beeldscherm worden gezet. Dit puntje kan in een bepaalde kleur worden uitgevoerd; indien geen kleur wordt gegeven, wordt de voorgrondkleur die op dat moment actief is, als kleur genomen.

De plaats op het beeldscherm kan als volgt worden bepaald:

Hoog oplossend vermogen: de linker bovenhoek van het beeldscherm krijgt positie 0,0. De rechter onderhoek krijgt positie 255,191. Tussen deze twee grenzen kan elk puntje worden aangesproken. Een schema ter verduidelijking:



Laag oplossend vermogen: de plaats op het beeldscherm wordt precies hetzelfde bepaald. Echter, op de puntjes die op het beeldscherm worden geplaatst, zijn bij een laag oplossend vermogen vier maal zo groot als bij een hoog oplossend vermogen in beide richtingen. Hierdoor geven bij

voorbeeld de locaties (0,0) tot en met (3,3) bij een laag oplossend vermogen dezelfde locatie voor een puntje aan.

Voorbeelden: (allemaal te onderbreken met CONTROL-STOP)

```
NEW
10 SCREEN 2
20 PSET (10,10)
30 PSET (30,30)
40 PSET (10,30)
50 PSET (30,10)
60 GOTO 60
RUN
```

Op het beeldscherm verschijnen vier puntjes als hoekpunten van een vierkant. Regel 60 voorkomt dat ongewenst weer een alfanumeriek scherm wordt geactiveerd en de tekening verdwijnt.

```
10 SCREEN 3
RUN
```

Alleen programmaregel 10 werd veranderd. Dezelfde punten worden afgebeeld; de punten zijn echter in beide richtingen vier maal zo groot.

```
NEW
10 SCREEN 2
20 FOR I=10 TO 30
30 FOR J=10 TO 30
40 PSET (I,J)
50 NEXT J,I
60 GOTO 60
RUN
```

Het vierkant waarvan in de eerdere voorbeelden alleen de hoekpunten werden afgebeeld, verschijnt nu in ingekleurde vorm doordat elk puntje binnen dit vierkant wordt geplaatst.

```
10 SCREEN 3
40 PSET (4*I,4*J)
RUN
```

Alleen de programmaregels 10 en 40 worden veranderd. Hetzelfde vierkant maar dan zestien maal vergroot (vier maal in elke richting) wordt afgebeeld. Merk op dat in programmaregel 40 steeds de locatie van de linker bovenhoek van het te plaatsen puntje wordt aangegeven.

```
10 SCREEN 2
RUN
```

Alleen programmaregel 10 werd veranderd. Het resultaat is dat nu een raster van puntjes wordt afgedrukt. Elk puntje geeft de linkerbovenhoek aan van een puntje dat zou zijn geplaatst bij een SCREEN 3-inrichting (voorlaatste voorbeeld).

```
10 SCREEN 3
40 PSET (4*I,4*J),15*RND(1)
RUN
```

Alleen programmaregels 10 en 40 werden veranderd. De punten zoals in het voorlaatste voorbeeld worden in willekeurige kleuren uitgevoerd.

```
10 SCREEN 2
RUN
```

Alleen programmaregel 10 werd veranderd. De punten zoals in het voorlaatste voorbeeld worden weer afgebeeld maar nu in diverse kleuren. De kleuren zijn niet helemaal willekeurig; in dit voorbeeld zijn de kleuren van de punten horizontaal twee aan twee gelijk.

De hoogste resolutie (oplossend vermogen) brengt met zich mee dat voor elk puntje niet elke kleur kan worden geselekteerd. De kleur wordt per groep van puntjes bepaald waarbij de laatst vastgestelde kleur geldig is. Kleuren worden per groep van acht naast elkaar liggende puntjes bepaald. De groepering is enigszins onduidelijk en is alleen goed te begrijpen wanneer men de exacte opbouw van het video-geheugen weet. Het is raadzaam om in de hoge resolutie kleuren met mate te gebruiken en de diverse kleuringen op enige afstand van elkaar te gebruiken. Misschien is het zelfs wel verstandig om, teneinde problemen te voorkomen, slechts twee kleuren (voorgrond en achtergrond) te gebruiken indien dat enigszins mogelijk is.

De locatie op scherm kan ook met behulp van het STEP sleutelwoord worden bepaald. In dat geval spreekt men over een relatieve locatie; de

beeldschermlocatie wordt gegeven ten opzichte van het laatste getekende punt. Bijvoorbeeld:

```
NEW
10 SCREEN 2
20 PSET (0,0)
30 FOR I=1 TO 20
40 PSET STEP (10,10)
50 NEXT I
60 GOTO 60
RUN
```

Op één lijn, steeds 10 puntjes horizontaal en vertikaal verderop, verschijnen puntjes op het beeldscherm. Programmaregel 20 zet het eerste puntje in de linker bovenhoek; programmaregel 40 zet vervolgens steeds (10,10) verderop een nieuw puntje neer. Steeds wordt een volgende locatie bepaald door de aangegeven stapgrootten in horizontale en vertikale richting bij de vorige locatie op te tellen.

```
NEW
10 SCREEN 2
20 PSET (10,10)
30 PSET STEP (20,20)
40 PSET STEP (-20,0)
50 PSET STEP (20,-20)
60 GOTO 10
RUN
```

Dit programma geeft precies hetzelfde resultaat als het eerste voorbeeld; het geeft de vier hoekpunten van een vierkant aan.

De tussen haakjes vermelde locaties dienen integere waarden te bevatten. Indien één van deze uitdrukkingen een gebroken waarde als uitkomt heeft, wordt deze waarde ontdaan van een decimale fractie (de decimalen worden verwaarloosd). Indien de uiteindelijke waarde van een uitdrukking dan kleiner is dan –32768 of groter is dan 32767, volgt een foutmelding.

Het is opmerkelijk dat er puntjes *buiten* het beeldscherm kunnen worden gezet. Deze mogelijkheid heeft een reële betekenis. Hierop wordt in de diverse beschrijvingen van de grafische kommando's verder ingegaan.

SLEUTELWOORD # PUT SPRITE

soort . . . . **KOMMANDO**

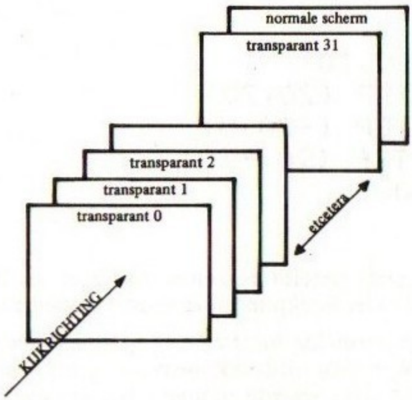
schrijfwijze

```
PUT SPRITE<TRANSPARANT NUMMER>,[<LOCA-
  TIE>][,[<KLEUR>][,<SPRITE NUMMER>]]\
  PUT SPRITE<...>,{,}
<LOCATIE>::=[STEP](<HORIZONTAAL>,<VER-
  TIKAAL>)
```

betekenis

Alvorens dit kommando te bestuderen, is het een noodzaak om SPRITE$, COLOR, SCREEN en PSET grondig te hebben bestudeerd.

Het beeldscherm van een MSX-computer kan men zich als volgt voorstellen:

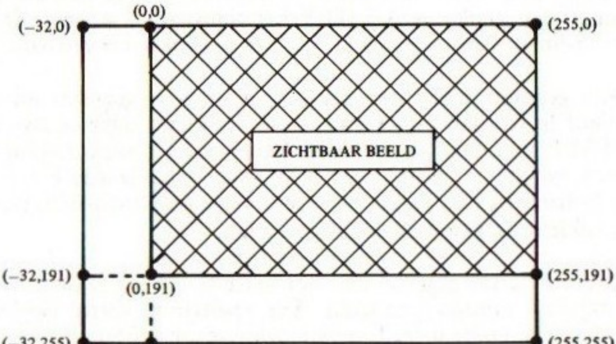


Het beeldscherm bestaat in feite uit een normaal (achtergrond)scherm met daarvoor 32 transparante schermen. Tot nu toe gebruikten we deze transparanten niet maar tekenden we slechts op het achtergrondscherm.
Met het PUT SPRITE kommando kan een sprite op een willekeurige

plaats in een willekeurige kleur op een willekeurig transparant worden afgebeeld. De volgende regels gelden:

- een sprite kan maar één kleur hebben
- er kan maar één sprite per transparant geplaatst worden
- indien twee sprites geheel of gedeeltelijk achter elkaar worden geplaatst, dan wordt ook visueel de achterste sprite achter de voorste geplaatst. De voorste sprite is dus volledig te zien terwijl de daar achter liggende sprite slechts gedeeltelijk of soms helemaal niet is te zien
- er kunnen maximaal 4 sprites tegelijk op één horizontale lijn staan. Indien meer dan 4 sprites op verschillende transparanten naast elkaar worden gezet, dan verdwijnen de achterste sprites visueel geheel of gedeeltelijk. Meer dan 4 sprites ONDER elkaar is geen probleem
- een sprite is op de niet getekende plaatsen (de witte ruitjes; zie SPRITE$) doorzichtig. Zo geeft het passeren van twee sprites altijd een visueel aanvaardbaar effect.

Sprites mogen zich binnen het transparant waarop ze zich bewegen, de volgende posities innemen:



Een sprite kan zich dus geheel of gedeeltelijk buiten het beeld bevinden.

Achter PUT SPRITE dienen we als eerste gegeven het transparantnummer aan te geven van het transparant waarop de sprite moet worden

geplaatst. Een eventueel in eerdere instantie op die transparant geplaatste sprite verdwijnt automatisch bij het plaatsen van de nieuwe transparant. Het transparantnummer wordt gegeven in een numerieke uitdrukking en mag niet kleiner zijn dan nul en niet groter dan 31. Een eventuele decimale fractie wordt verwaarloosd.

Als tweede gegeven achter PUT SPRITE kan de locatie van de betreffende sprite worden gegeven. Deze locatie dient aan dezelfde eisen te voldoen als de locatie zoals voorkomend in het PSET-kommando en kan dus zowel absoluut als relatief zijn. Indien de locatie wordt weggelaten, wordt de laatste gespecificeerde locatie voor dat transparant genomen. Indien nog niet eerder een locatie voor de betreffende sprite werd gespecificeerd, wordt locatie (–32,192) aangenomen.
Pas op: Indien een relatieve locatie wordt opgegeven (dus een STEP (...,...)) dan wordt deze relatieve locatie uitgevoerd *ten opzichte van de laatst gegeven locatie*. Op welk grafisch kommando deze laatste locatie van toepassing was, is dan niet van belang. De opgegeven relatieve locatie is in ieder geval NOOIT de locatie ten opzichte van de vorige locatie *voor die betreffende sprite*. Dit is een kleine fout in MSX-basic.

Indien de y-coördinaat van een sprite (de vertikale positionering) gelijk is aan 208, dan heeft dat tot effect dat alle sprites die op een transparant zijn geplaatst ACHTER het transparant waarop de sprite met y-coördinaat 208 is geplaatst, van het beeldscherm verdwijnen.

Als derde gegeven dient een kleur te worden opgegeven waarin de sprite dient te worden uitgevoerd. Dezelfde kleurkodering als gegeven bij het PSET-kommando geldt hier. Indien geen kleurkodering wordt opgegeven, wordt de laatste voor deze sprite gekodeerde kleur aangenomen. Indien nog niet eerder een kleur werd gespecificeerd, wordt de voorgrondkleur als geldende kleur genomen.

Als vierde en laatste gegeven kan het betreffende sprite (de betreffende spelfiguur) worden genoemd. Dit spelfiguur dient eerder met SPRITE$ te zijn gedefinieerd. Indien dit gegeven achterwege wordt gelaten, dan neemt het MSX-basic aan dat het nummer van het spelfiguur gelijk is aan het nummer van het transparant.

Hieronder volgen enkele voorbeelden. Het is noodzakelijk om veel met sprites te oefenen alvorens aan wat meer serieus programmeerwerk te beginnen.

Voorbeeld 1, definitie van een 8 bij 8 sprite. Dit voorbeeld wordt bij

de behandeling van SPRITE$ uitvoerig behandeld.

```
NEW
10 SPRITE$(0)=CHR$(24)+CHR$(60)+CHR$(60)
CHR$(86)+CHR$(126)+CHR$(126)+CHR$(230)+
CHR$(124)
```

Met dit kommando is spelfiguur nummer 0 bepaald. Het spelfiguur is gelijk aan het eerder bij SPRITE$ bepaalde figuur met dit verschil dat er decimale en geen binaire konstanten zijn gebruikt.
Om geen foutmelding te krijgen, dienen we voorafgaand aan dit kommando eerst de juiste beeldscherminstelling te realiseren:

```
5 SCREEN 2,1
```

Gekozen werd (zie SCREEN) voor een grafisch beeldscherm met een sprite-grootte van 8 bij 8 beeldpunten en een vier maal zo grote weergave van deze sprite.
Vervolgens gaan we de sprite op het beeldscherm plaatsen om te kijken hoe de sprite er nu werkelijk uit ziet:

```
20 PUT SPRITE 0,(100,100),15,0
30 GOTO 30
RUN
```

Op regel 20 werd de spelfiguur op transparant 0 geplaatst en wel op locatie (100,100). Als kleur werd voor wit gekozen en, in dit geval ten overvloede, werd voor sprite nummer 0 (SPRITE$(0)) gekozen. Op het beeldscherm zien we een soort 'spookje', bruikbaar in diverse reactiespelletjes. Regel 30 is een eeuwigdurende loop om het grafische beeld vast te houden. Onderbreek met CONTROL-STOP.
Vervolgens laten we een tweede spookfiguurtje in een andere kleur achter dit eerste spookje langs bewegen:

```
30 FOR I=0 TO 200
40 PUT SPRITE 1,(I,104),10,0
50 NEXT I
60 GOTO 30
RUN
```

Snel zien we een tweede spookje (donkergeel) achter het eerste spookje langs gaan, zich iets lager bewegend. Om deze beweging goed volg-

baar te maken, kunnen we eventueel de volgende wijziging intoetsen:

```
30 FOR I=0 TO 200 STEP .1
```

Een derde spookje kunnen we weer achter het tweede langs laten gaan:

```
45 PUT SPRITE 2,(104,I),8,0
RUN
```

Om een grote snelheid in de beweging te realiseren, kunnen we bijvoorbeeld de volgende verandering aanbrengen:

```
30 FOR I=0 TO 200 STEP 5
```

Steeds zien we dat in de for-next-lus op de transparanten nummer 1 en 2 de figuren op een andere plaats worden afgebeeld. De eerder op deze transparant geplaatste afbeelding verdwijnt hierbij; er kan maar één sprite tegelijk op een transparant worden afgebeeld.

Vervolgens kunnen we deze figuurtjes zich voor een bepaalde achtergrond laten bewegen:

```
11 LET A=RND(-TIME)
12 FOR I=1 TO 20
13 CIRCLE (RND(1)*255,RND(1)*191),
RND(1)*80
14 NEXT I
15 FOR I=1 TO 10
16 PAINT (RND(1)*255,RND(1)*191)
17 NEXT I
RUN
```

De figuurtjes verplaatsen zich nu voor een achtergrond van willekeurige cirkels waarvan gedeelten al of niet zijn ingekleurd. Zie de behandeling van RND en TIME voor een optimaal begrip. Elke keer dat het programma wordt opgestart, is de achtergrond weer wat anders.

Door op transparant nummer 1 de y-coördinaat 208 te gebruiken, kunnen we het spookje op transparant nummer 2 bijvoorbeeld laten knipperen:

```
40 PUT SPRITE 1,(0,208+Q)
41 LET Q=NOT(Q)
```

Regel 41 zorgt ervoor dat variabele Q afwisselend de waarde 0 en –1 krijgt. Hierdoor wordt op regel 40 afwisselend een figuurtje wel en niet op de y-coördinaat 208 van transparant 1 geplaatst. Hierdoor wordt het figuurtje op transparant 2 afwisselen wél en niet zichtbaar waardoor een knipperend effect ontstaat.

Een spelfiguur kan men laten bewegen door bijvoorbeeld afwisselend andere, iets gewijzigde sprites op dezelfde locatie op eenzelfde transparant af te beelden. Het volgende voorbeeld laat een zich schuin over het beeld verplaatsend en pulserend cirkeltje zien:

```
NEW
10 SCREEN 2,1
20 SPRITE$(0)=CHR$(126)+STRING$(6,129)+
CHR$(126)
30 SPRITE$(1)=CHR$(0)+CHR$(60)+
STRING$(4,66)+CHR$(60)+CHR$(0)
40 SPRITE$(2)=STRING$(2,0)+CHR$(24)+
STRING$(2,36)+CHR$(24)+STRING$(2,0)
50 SPRITE$(3)=STRING$(3,0)+STRING$(2,24)
+STRING$(3,0)
60 FOR I=0 TO 3.9 STEP 0.05
70 PUT SPRITE 0,STEP(1,1),15,I
80 NEXT I
90 GOTO 60
RUN
```

Om het spelfiguur zich te laten verplaatsen, werd gebruik gemaakt van de STEP-mogelijkheid. Om het spelfiguur continu te laten veranderen, werd gebruik gemaakt van vier apart gedefinieerde sprites die achter elkaar op hoegenaamd dezelfde locatie en op dezelfde transparant worden geplaatst. Het gebruik van de STEP op regel 60 dient om de verandering van het spelfiguur niet te snel te laten geschieden; probeer de stepwaarde maar eens te veranderen.

Het voorbeeld laat een nog niet eerder genoemd verschijnsel zien; een sprite die aan de ene kant in verband met de locatie geheel of gedeel-

telijk buiten de toegestane grenzen treedt, wordt aan de tegenovergestelde kant weer (geheel of gedeeltelijk) in beeld teruggebracht. Dit gebeurt door bij/van een buiten de toegestane grenzen tredende coördinaat net zo vaak de waarde 256 op te tellen/af te trekken totdat de betreffende coördinaat binnen de toegestane grenzen valt.
De programmaregels:

```
10 PUT SPRITE 9,(100,100),15,1
10 PUT SPRITE 9,(356,356),15,1
10 PUT SPRITE 9,(-156,-156),15,1
```

hebben dus allemaal precies hetzelfde effekt.

Tot slot volgt hier een programma dat de eerder in de behandeling van SPRITE$ ontworpen 16 bij 16 sprite gebruikt:

```
NEW
10 COLOR 2,1,1:KEY OFF:LET A=RND(-TIME)
20 RESTORE 100:LET A$="":FOR I=1 TO 32
30 READ A:LET A$=A$+CHR$(A):NEXT I
40 SCREEN 1,3:SPRITE$(0)=A$
50 FOR I=1 TO 31
60 PUT SPRITE I,(RND(1)*255,RND(1)*
255),RND(1)*15,0
70 PRINT "MSX-BASIC *** ";
80 NEXT I
90 GOTO 50
100 DATA 3,15,15,31,27,25,31,31,30,59,
124,124,63,31,3,1
110 DATA 224,240,248,248,184,152,248,
248,124,220,30,60,120
120 DATA 240,192,128
RUN
```

Op beeld wordt continu de tekst MSX-BASIC *** afgedrukt. Op de voorgrond worden op willekeurige posities in willekeurige kleuren, spookfiguurtjes voor- en achter elkaar afgebeeld. Uit dit programma kunnen we concluderen dat het gebruik van sprites is toegestaan bij SCREEN 1 (alfanumeriek), SCREEN 2 (grafisch) en SCREEN 3 (gra-

fisch). Hierdoor is het op eenvoudige wijze mogelijk om sprites zich over een tekst te laten bewegen. Ook zien we dat diverse malen een sprite slechts gedeeltelijk wordt afgedrukt; er staan op dat moment méér dan vier sprites op dezelfde regel. Ook zien we een enkele keer dat er plotseling een aantal sprites geheel verdwijnen. In dat geval heeft één van de sprites gedurende een for-next-loop een y-coördinaat 208 aangenomen.

SLEUTELWOORD **REM**

---

soort . . . . **KOMMANDO**

schrijfwijze

```
REM<COMMENTAAR>
```

betekenis

Het REM-kommando dient slechts ter opname van kommentaar in het programma. Het programma zal bij uitvoering de REM-kommando's overslaan.

Pas op: Het REM-kommando dient altijd als LAATSTE kommando in een programmaregel te worden opgenomen. Voorbeeld:

```
NEW
10 REM DIT PROGRAMMA DOET NIETS
20 STOP
RUN
Break in 20
Ok
```

Indien kommentaar NAAST kommando's dient te worden opgenomen, kan ook het enkele aanhalingsteken (') worden gebruikt. Ook kommentaar, opgenomen achter een ('), dient als laatste in een programmaregel te zijn opgenomen. Voorbeeld:

```
NEW
```

```
10 REM VOORBEELDPROGRAMMA
20 INPUT A:REM DE WAARDE VAN A WORDT
HIER INGEGEVEN
30 PRINT A'DE WAARDE VAN A WORDT HIER
AFGEDRUKT
RUN
? 12.5
 12.5
Ok
```

SLEUTELWOORD RENUM

---

soort .... **KOMMANDO**

schrijfwijze

```
RENUM[<NIEUW REGELNUMMER>][,[<OUD RE-
  GELNUMMER>][,<STAPGROOTTE>]]
```

betekenis

Met RENUM kunnen programmaregels van een nieuw regelnummer worden voorzien. Dit kan bijvoorbeeld praktisch zijn wanneer u programmaregels wilt tussenvoegen maar er tussen de twee programmaregels niet meer voldoende plaats is.

Indien u slechts RENUM intoetst, dan wordt het volledige programma hernummerd. Het eerste regelnummer is regelnummer 10 terwijl de volgende regels steeds een regelnummer krijgen dat 10 hoger ligt.
Indien u RENUM ingeeft, gevolgd door een regelnummer dan heeft dat tot gevolg dat de eerste programmaregel dit nummer krijgt en dat alle volgende programmaregels steeds een nummer krijgen dat 10 hoger ligt.
Indien u RENUM ingeeft, gevolgd door een regelnummer, een komma en weer een regelnummer, dan heeft dat tot gevolg dat het programma vanaf het tweede ingegeven regelnummer wordt hernummerd. Het eerste gedeelte blijft dus onveranderd. Het eerste te hernummeren

regelnummer krijgt het eerste ingegeven regelnummer. Indien het eerste regelnummer wordt overgeslagen (dus alleen een komma en het tweede regelnummer) dan krijgt het eerste te hernummeren regelnummer het nummer 10 mee. In beide gevallen wordt er weer met stappen van 10 opgenummerd.
Indien u RENUM intoetst, gevolgd door een regelnummer, een komma, een regelnummer, een komma en een stapgrootte dan heeft dat tot effekt dat behalve dat er hernummerd wordt zoals zojuist beschreven daarbij de ingegeven stapgrootte wordt aangenomen in plaats van de normaal gehanteerde stapgrootte van 10. Indien een RENUM wordt gegeven met daarbij alleen de stapgrootte gedefinieerd (RENUM ,, stapgrootte), dan wordt het programma geheel hernummerd waarbij het eerste regelnummer gelijk is aan 10. De volgende regels liggen steeds een stapgrootte hoger.

Voorbeeld:

```
NEW
10 REM
20 REM
30 REM
RENUM 1,,1
Ok
LIST
1 REM
2 REM
3 REM
Ok
RENUM
LIST
10 REM
20 REM
30 REM
Ok
RENUM 1000,20,5
Ok
LIST
10 REM
1000 REM
```

```
1005 REM
Ok
```

RENUM werkt ALLE regelnummers naar behoren bij. Zo ook bijvoorbeeld de regelnummers die in een GOTO of een GOSUB of een ON... GOTO kommando vermeld staan.

Indien RENUM in een programmaregel is opgenomen, dan probeert MSX-basic ten onrechte ook om de regelnummers en de stapgrootte die eventueel achter het RENUM-kommando zijn vermeld, te hernummeren. Het één en ander kan leiden tot foutmeldingen en curieuze situaties. Het gebruik van een RENUM in een programmaregel is echter meestal vrij zinloos.

SLEUTELWOORD # RIGHT$

---

soort . . . . **A-FUNKTIE**

schrijfwijze

```
RIGHT$(<BASISSTRING>,<AANTAL TEKENS>)
```

betekenis

Deze funktie geeft van een alfanumerieke uitdrukking het rechtergedeelte als resultaat. Hoe groot dit rechtergedeelte is, wordt bepaald door het opgegeven aantal tekens. Bijvoorbeeld:

```
PRINT RIGHT$("MSX-BASIC",5)
BASIC
Ok
```

Het aantal tekens mag niet kleiner zijn dan nul en niet groter dan 255. Een eventuele decimale fractie wordt verwaarloosd.

Nog een voorbeeld:

```
NEW
10 LINE INPUT "HOE HEET U ?:";NAAM$
```

```
20 INPUT "HOEVEEL LETTERS HEEFT UW
ACHTERNAAM";AANTAL
30 PRINT "HALLO MENEER/MEVROUW ";RIGHT$
(NAAM$,AANTAL);"!"
RUN
HOE HEET U ?:JOHAN KARDINAAL
HOEVEEL LETTERS HEEFT UW ACHTERNAAM ? 9
HALLO MENEER/MEVROUW KARDINAAL!
Ok
```

SLEUTELWOORD **RND**

---

soort . . . . **N-FUNKTIE**

schrijfwijze

RND(<N>)

betekenis

Deze funktie geeft als resultaat een toevalsgetal. Dit getal is minimaal gelijk aan nul maar altijd kleiner dan 1.
De tussen haakjes opgenomen numerieke uitdrukking heeft de volgende betekenis:

- kleiner dan nul: het eerste toevalsgetal uit een door de numerieke uitdrukking bepaalde reeks wordt berekend en als resultaat gegeven. Voortzetting van de reeks toevalsgetallen dient te geschieden door bij elke volgende RND een uitdrukking met positieve waarde te gebruiken
- groter dan nul: het volgende toevalsgetal uit een reeds aangevangen reeks wordt gepresenteerd. Indien de reeks niet eerder werd aangevangen, wordt het getal 0.59521943994623 als eerste uit de reeks gegenereerd
- gelijk aan nul: het laatste toevalsgetal wordt herhaald.

Indien nog niet eerder een toevalsgetal werd bepaald, wordt het getal 0.40649651372358 gegenereerd

De term toevalsgetal is niet correct. De toevalsgetallen worden door de computer BEREKEND waarbij een zo goed mogelijke verdeling van de getallen wordt betracht. Deze berekening is echter zo complex dat een volgend getal uit een reeks niet voorspelbaar is tenzij de reeks reeds eerder werd bestudeerd.

Een voorbeeld: een speltoepassing:

```
NEW
10 REM DOBBELSTEEN
20 INPUT "GEEF EEN WAARDE IN ";A
25 OGEN=RND(-A)
30 PRINT "GEEF EEN TOETS IN"
40 K$=INKEY$:IF K$="" THEN 40
50 OGEN=INT(6*RND(1)+1)
60 PRINT "IK GOOIDE EEN";OGEN
70 GOTO 30
RUN
GEEF EEN WAARDE IN ? 11
GEEF EEN TOETS IN
IK GOOIDE EEN 5
GEEF EEN TOETS IN
IK GOOIDE EEN 5
GEEF EEN TOETS IN
IK GOOIDE EEN 3
```

(etcetera; onderbreken met CONTROL-STOP)

Op regel 25 wordt naar aanleiding van een ingegeven getal een reeks toevalsgetallen aangevangen. Op regel 50 wordt dan steeds de volgende uit de reeks bepaald. Met dit toevalsgetal wordt een berekening uitgevoerd, zodanig dat het eindresultaat een willekeurig geheel getal is, gelijk aan 1, 2, 3, 4, 5, of 6.

Zie ook de behandeling van TIME.

SLEUTELWOORD # RUN

soort . . . . **KOMMANDO**

schrijfwijze

```
RUN

[<REGELNUMMER>[{<KARAKTER>}\<CIJFER><...>]]
[        "<PROGRAMMANAAM>"[,R]             ]
```

betekenis

Het RUN-kommando is het meest eenvoudige maar ook het meest belangrijke kommando dat MSX-basic kent. Met dit kommando kunnen we een in het werkgeheugen aanwezig MSX-basic programma activeren.

Met een RUN activeren we een programma:

```
NEW
10 PRINT "TEST 1"
20 PRINT "TEST 2"
RUN
TEST 1
TEST 2
Ok
```

Indien we achter het RUN-kommando een regelnummer opnemen, zal het programma vanaf dit regelnummer worden geactiveerd. Voorbeeld:

```
NEW
10 PRINT "TEST 1"
20 PRINT "TEST 2"
RUN 20
TEST 2
Ok
```

RUN sluit alle eventueel openstaande kanalen en maakt daarbij alle variabelen schoon. Merk op dat RUN standaard onder funktietoetsen 5 en 10 aanwezig is in twee verschillende uitvoeringen.

SLEUTELWOORD # SCREEN

soort .... **KOMMANDO**

schrijfwijze

```
SCREEN[<INSTELLING>][,[<SPRITEGROOTTE>][,
  [<KLIK>][,[<CASSETTE-SCHRIJFSNELHEID>]
  [,<PRINTERTYPE>]]]]\SCREEN[<...>,{,}]
```

betekenis

Met dit kommando kunnen diverse instellingen van de computer worden bepaald, hoofdzakelijk in verband met het beeldscherm.

Alle numerieke uitdrukkingen die binnen het SCREEN-kommando worden gebruikt, worden indien zij gebroken waarden hebben, herleid naar het grootste gehele getal kleiner dan deze gebroken waarde.

1) Instelling: de waarde van de numerieke uitdrukking die de instelling bepaalt, mag gelijk zijn aan 0, 1, 2 of 3. Deze waarden hebben de volgende betekenissen:

SCREEN 0 een alfanumeriek scherm van maximaal 40 karakters bij 24 regels wordt ingericht. Deze scherminrichting staat geen grafisch gebruik toe

SCREEN 1 wederom wordt een alfanumeriek scherm ingericht. Ditmaal zijn de afmetingen maximaal 32 bij 24 regels. Ook deze beeldscherminrichting staat geen grafisch gebruik toe

SCREEN 2 een grafisch scherm wordt ingericht met een hoog oplossend vermogen (er kan gedetailleerd op worden getekend). Het kleurgebruik is enigszins aan beperkingen onderhevig. Deze beeldscherminrichting staat geen

alfanumeriek gebruik toe

SCREEN 3 een grafisch scherm met een laag oplossend vermogen (er kunnen alleen vrij grove tekeningen op worden gemaakt) en een onbeperkte mogelijkheid tot gebruik van 15 kleuren wordt ingericht. Ook deze beeldscherminstelling staat geen alfanumeriek gebruik toe.

Er zijn twee verschillende grafische inrichtingen van het beeldscherm mogelijk, namelijk een inrichting met een hoog oplossend vermogen en een inrichting met een laag oplossend vermogen. Onderstaande tabel geeft een beeld van het oplossende vermogen:

	beeldschermindeling	
	horizontaal	vertikaal
SCREEN 2	256 puntjes	192 puntjes
SCREEN 3	64 puntjes	48 puntjes

Een instelling van het beeldscherm op bovenstaande wijze heeft altijd tot gevolg dat het beeldscherm wordt schoongemaakt.

2) Spritegrootte: de waarde van de numerieke uitdrukking die de spritegrootte bepaalt, mag gelijk zijn aan 0, 1, 2 of 3. Deze waarden hebben de volgende betekenissen:

SCREEN ,0 de sprite-grootte is 8 bij 8 stippen

SCREEN ,1 de sprite-grootte is 8 bij 8 stippen; sprites worden vergroot weergegeven

SCREEN ,2 de sprite-grootte is 16 bij 16 stippen

SCREEN ,3 de sprite-grootte is 16 bij 16 stippen; sprites worden vergroot weergegeven.

De zin van deze inrichting van de sprite-grootte wordt pas duidelijk bij de behandeling van de SPRITE-sleutelwoorden. De vergroting geschiedt altijd met een faktor 4 (in lengte en breedte 2x zo groot).

3) Klik: de waarde van deze numerieke uitdrukking is minimaal 0 en maximaal 255. Deze waarde heeft de volgende betekenis:

SCREEN ,,0 de klik die normaal bij de toetsaanslag hoorbaar is, verdwijnt

SCREEN ,,1 bij invulling van een 1 of elke andere toegestane waarde groter dan nul is het effect dat de klik weer wordt geactiveerd.

4) Cassette schrijfsnelheid: de waarde van deze numerieke uitdrukking mag gelijk zijn aan 1 of 2 en heeft de volgende betekenis:

`SCREEN ,,,1` het schrijven van cassetteband gebeurt met een snelheid van 1200 baud (ongeveer 150 tekens per seconde)

`SCREEN ,,,2` het schrijven van de cassetteband gebeurt met een snelheid van 2400 baud (ongeveer 300 tekens per seconde). Deze instelling mag uitsluitend worden gebruikt indien zowel de cassetterecorder als het cassettebandje van uitmuntende kwaliteit zijn. In een ander geval is de kans op optreden van lees- en schrijffouten erg groot.

5) Printer type: de waarde van deze numerieke uitdrukking is minimaal gelijk aan nul en maximaal gelijk aan 255. Deze waarde heeft de volgende betekenis:

`SCREEN ,,,,0` een printer met mogelijkheid tot het afdrukken van de MSX grafische symbolen is aangesloten

`SCREEN ,,,,1` elke toegestane waarde groter dan nul geeft aan dat een printer is aangesloten zonder de mogelijkheid tot het afdrukken van de MSX grafische symbolen. In voorkomende gevallen worden in plaats van de symbolen dan spaties afgedrukt.

enkele voorbeelden:

```
SCREEN 3,0,1,2,1
```

Gekozen werd voor een grafische scherminrichting met laag oplossend vermogen en maximaal kleurgebruik. De sprite-grootte werd bepaald op 8 bij 8 stippen onvergroot, de klik bij toetsinslag werd aangezet, de cassetteschrijfsnelheid werd maximaal ingesteld en de computer werd medegedeeld dat de aangesloten printer niet in staat is om MSX grafische tekens te reproduceren.

```
SCREEN ,1,,1
```

De sprite-grootte werd veranderd naar 8 bij 8 stippen vergroot terwijl de cassette-schrijfsnelheid naar 1200 baud werd teruggebracht. Voor het overige werden geen wijzigingen aan de instelling doorgevoerd.

Voor alle met SCREEN geformuleerde instellingen geldt dat een eventuele decimale fractie wordt verwaarloosd.

Merk op dat wanneer een programma of een direkt kommando is afgerond en dat de MSX-computer weer de Ok-melding moet gaan afdrukken, of wanneer een INPUT vanaf het toetsenbord moet worden uitgevoerd of een foutmelding moet worden gegeven, automatisch een eventueel actieve grafische schermindeling wordt gedeactiveerd en dat de laatste geactiveerde alfanumerieke scherminstelling weer wordt geactiveerd. Deze actie is logisch; om in MSX-basic te kunnen programmeren of om een ingave te kunnen plegen is natuurlijk een alfanumerieke schermindeling noodzakelijk. Een PRINT-kommando tast echter een grafische instelling niet aan; bij een grafisch beeldscherm wordt de PRINT op het beeldscherm gewoon verwaarloosd.

Pas bij behandeling van de diverse grafische mogelijkheden komt de zin van het SCREEN-sleutelwoord volledig tot uiting.

N.B.: Bij het opstarten van de computer worden standaard de laagste instellingen uitgevoerd.

SLEUTELWOORD **SGN**

---

soort . . . . **N-FUNKTIE**

schrijfwijze

SGN(<N>)

betekenis

Deze funktie kan drie resultaten geven:

de waarde 1 wanneer de waarde van de tussen haakjes vermelde

numerieke uitdrukking positief (groter dan nul) is

de waarde 0 — wanneer de waarde van de tussen haakjes vermelde numerieke uitdrukking gelijk is aan nul

de waarde –1 — wanneer de waarde van de tussen haakjes vermelde numerieke uitdrukking negatief (kleiner dan nul) is.

Voorbeeld:

```
NEW
10 LET A=-123.4:LET B=0:LET C=1E33
20 PRINT SGN(A);SGN(B);SGN(C)
RUN
-1  0  1
Ok
```

SLEUTELWOORD SIN

soort . . . . **N-FUNKTIE**

schrijfwijze

```
SIN(<HOEK>)
```

betekenis

Deze funktie geeft als resultaat de sinus van de waarde van de tussen haakjes vermelde uitdrukking. Deze waarde wordt beschouwd als de uitdrukking van een hoek in radialen. Het resultaat ligt uiteraard altijd tussen –1 en 1.

Alhoewel elke waarde voor de hoek is toegestaan, is het in verband met de precisie van berekenen raadzaam om niet te grote hoekmaten te gebruiken.

Voorbeeld:

```
NEW
10 FOR I=0 TO 90
20 LET HOEK=I/180*3.1415926535
30 PRINT I;":";SIN(HOEK)
40 NEXT I
RUN
```

(een sinustabel verschijnt op het beeldscherm. Merk op dan op regel 20 de hoek van graden naar radialen wordt geconverteerd)

SLEUTELWOORD # SPACE$

---

soort . . . . **A-FUNKTIE**

schrijfwijze

```
SPACE$(<AANTAL SPATIES>)
<AANTAL SPATIES>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Deze funktie geeft als resultaat een aantal spaties. Het aantal wordt bepaald door de numerieke uitdrukking, opgenomen tussen de haakjes. Indien de uitkomst van deze numerieke bewerking een gebroken getal is, wordt de grootste waarde, kleiner dan dat getal als geldige waarde genomen.

Voorbeeld:

```
NEW
10 FOR I=0 TO 9
20 PRINT SPACE$(I);"MSX"
30 NEXT I:STOP
RUN
MSX
 MSX
```

```
 MSX
  MSX
   MSX
    MSX
     MSX
      MSX
       MSX
        MSX
Break in 30
Ok
```

SLEUTELWOORD SPRITE$

soort . . . . **SYSTEEMVARIABELE**

schrijfwijze

```
SPRITE$(<SPRITE NUMMER>)
```

betekenis

Het is noodzakelijk dat het SCREEN-kommando reeds grondig is bestudeerd.

Het MSX-basic voorziet in vele kommando's die het schrijven van spelprogramma's in BASIC bijzonder goed ondersteunen. Gebruik van de sleutelwoorden SPRITE, SPRITE$, PUT SPRITE en ON SPRITE GOSUB maken het mogelijk om:

- 32 verschillende vaste figuren (sprites) onafhankelijk van elkaar achter elkaar langs op het beeldscherm zich te laten verplaatsen;
- deze sprites of spelfiguren zelf te ontwerpen;

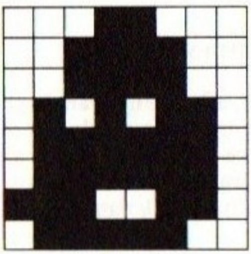
Spelprogramma's met bewegende beelden (packman-achtige spelletjes, aanvals- en schietspelletjes en dergelijke) kunnen in MSX-basic voor het eerst worden geprogrammeerd zonder dat kennis van machinetaal of assembler nodig is en met behoud van de benodigde snelheid. Daarbij zijn uiteraard ook wat serieuzere toepassingen bijzonder goed denkbaar.

Met SPRITE$ kunnen maximaal 64 (indien de sprite-grootte in het SCREEN-kommando op 2 of 3 is gezet) of 256 (indien de sprite-grootte in het SCREEN-kommando op 0 of 1 is gezet) spelfiguren onafhankelijk van elkaar worden ontworpen. Het ontwerpen van een sprite kan als volgt geschieden:

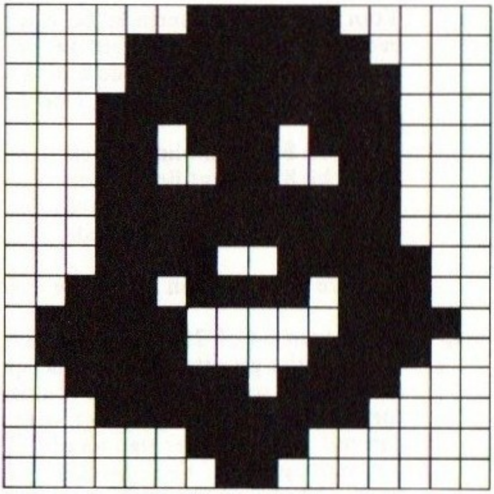
stap 1 beslis welke sprite-grootte gaat worden gebruikt. Is deze 8 bij 8 beeldpunten (sprite-grootte 0 of 1 in het SCREEN-kommando) of is deze 16 bij 16 beeldpunten (spritegrootte 2 of 3 in het SCREEN-kommando)

stap 2 maak een ontwerpblad. Dit kan het beste worden gedaan door een stuk ruitjespapier te nemen en daarop een vierkant van 8 bij 8 ruitjes of 16 bij 16 ruitjes af te bakenen, afhankelijk van de gewenste sprite-grootte.

stap 3 ontwerp binnen dit vierkant het gewenste figuur door ruitjes in te kleuren of wit te laten. Een voorbeeld 8 bij 8 en 16 bij 16:

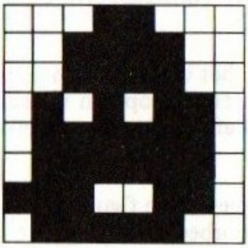


8x8 sprite



16x16 sprite

stap 4 digitaliseer de sprite. Voor een 8 bij 8 sprite is dat het gemakkelijkst. De ingekleurde ruitjes dienen voorgesteld te worden met het cijfer 1; de witte ruitjes met het cijfer 0. Zo krijgen we acht getallen van acht cijfers, allemaal nullen of enen. Voor deze acht cijfers zetten we "&B" waardoor we plotseling acht binaire konstanten hebben verkregen. Bijvoorbeeld:



8x8 sprite

```
&B 0 0 0 1 1 0 0 0
&B 0 0 1 1 1 1 0 0
&B 0 0 1 1 1 1 0 0
&B 0 1 0 1 0 1 1 0
&B 0 1 1 1 1 1 1 0
&B 0 1 1 1 1 1 1 0
&B 1 1 1 0 0 1 1 0
&B 0 1 1 1 1 1 0 0
```

digitalisatie

Voor een 16 bij 16 sprite is dat wat moeilijker. Verdeel hiertoe eerst het 16 bij 16 vierkant in vier 8 bij 8 vierkanten. Deze kunnen op de manier van de 8 bij 8 sprites weer worden gedigitaliseerd (in cijfers omgezet). Doe dit in de volgorde:

1: 8 bij 8 vierkant linksboven
2: 8 bij 8 vierkant linksonder
3: 8 bij 8 vierkant rechtsboven
4: 8 bij 8 vierkant rechtsonder

Op deze wijze worden 32 binaire kontanten verkregen.

stap 5 we kunnen 64 of 256 sprites definieren, afhankelijk van de gekozen sprite-grootte. Beslis welk sprite-nummer de door ons ontworpen sprite krijgt. De numerieke uitdrukking die als sprite-nummer wordt gebruikt, mag in waarde niet kleiner zijn dan nul en niet groter dan 63 of 255, afhankelijk van de gekozen sprite-grootte. Een eventuele decimale fractie wordt verwaarloosd. In ons voorbeeld zullen we sprite nummer 6 gaan definiëren.

stap 6 leg de sprite vast door SPRITE$(...) gelijk te stellen aan CHR$ (eerste binaire konstante)+CHR$(tweede binaire konstante)+ CHR$(derde...) etcetera totdat alle konstanten zijn opgebruikt. Bijvoorbeeld:

```
NEW
10 SPRITE$(6)=CHR$(&B00011000)+
CHR$(&B00111100)+CHR$(B00111100)+
CHR$(&B01010110)+CHR$(&B01111110)+
CHR$(&B01111110)+CHR$(&B11100110)+
CHR$(&B01111100)
```

Voor een 16 bij 16 sprite kan dit niet daar de programmaregel dan onvermijdelijk groter wordt dan 255 tekens, hetgeen verboden is in MSX-basic. Om dit te voorkomen, dienen we een methode te gebruiken die kortere programmaregels toestaat. Omdat deze methode veel minder ruimte inneemt dan de eerste methode, is deze ook in tweede instantie voor de 8 bij 8 sprites aan te raden.
Allereerst dienen we alle binaire konstanten te converteren naar decimale konstanten. Dit laten we de computer doen en wel op de volgende manier:
Voor elke binaire konstante tikken we:

`PRINT &B...` (de juiste binaire konstante invullen)

De computer antwoordt onmiddellijk met de decimale waarde van deze konstante die nooit meer dan drie cijfers groot is. Bijvoorbeeld:

```
PRINT &B11111111
 255
Ok
```

Voor elke binaire konstante laten we de computer de decimale waarde uitrekenen die we dan stuk voor stuk opschrijven. Uiteindelijk zetten we deze getallen in één of meer DATA-kommando's onder elkaar. Bijvoorbeeld (zie de eerder ontworpen 16 bij 16 sprite). (Zie vervolg tekst de volgende pagina.)

Indien met SPRITE$ een sprite wordt gedefinieerd, dient eerst de sprite-grootte te zijn bepaald (met een SCREEN). Indien dit niet gebeurt, volgt een foutmelding of worden (bij omschakelen van alleen de sprite-grootte) de sprites weer gewist.

N.B.: Het sleutelwoord LET mag niet voor SPRITE$ worden gebruikt.

Zie het PUT SPRITE kommando voor gebruik van ontworpen sprites.

00000011	11100000	3	224
00001111	11110000	15	240
00001111	11111000	15	248
00011111	11111000	31	248
00011011	10111000	27	184
00011001	10011000	25	152
00011111	11111000	31	248
00011111	11111000	31	248
00011110	01111100	30	124
00111011	11011100	59	220
01111100	00011110	124	30
01111100	00111100	124	60
00111111	01111000	63	120
00011111	11110000	31	240
00000011	11000000	3	192
00000001	10000000	1	128

pijl = volgorde voor DATA

digitalisering van de 16 x 16 sprite (zie tekening)

de corresponderende decimale waarden

Uiteindelijk resultaat: de DATA-regel(s):

```
NEW
100 DATA 3,15,15,31,27,25,31,31,30,
59,124,124,63,31,3,1
110 DATA 224,240,248,248,184,152,
248,248,124,220,30,60,120
120 DATA 240,192,128
```

Vervolgens zetten we de sprite eerst in een gewone alfanumerieke variabele op een wijze zoals deze:

```
10 RESTORE 100:LET A$="":FOR I=1 TO 32
20 READ A:LET A$=A$+CHR$(A):NEXT I
```

Als laatste actie bepalen we de 16 bij 16 sprite definitief, bijvoorbeeld door het kommando:

```
30 SCREEN 2,2:SPRITE$(6)=A$
```

SLEUTELWOORD SQR

soort . . . . **N-FUNKTIE**

schrijfwijze

```
SQR(<N>)
```

betekenis

Deze funktie geeft als resultaat de vierkantswortel van de waarde van de uitdrukking die tussen haakjes staat. Uiteraard mag deze waarde niet negatief zijn.

Voorbeeld:

```
NEW
10 INPUT "GETAL ";A
20 PRINT "DE VIERKANTSWORTEL"
30 PRINT "IS";SQR(A):GOTO 10
RUN
GETAL ? 3
DE VIERKANTSWORTEL
IS 1.7320508075688
GETAL ? 9
DE VIERKANTSWORTEL
IS 3
```

```
GETAL ? 1024
DE VIERKANTSWORTEL
IS 32
GETAL ? -1
DE VIERKANTSWORTEL
IS
Illegal function call in 30
Ok
```

SLEUTELWOORD **STOP**

---

soort . . . . **KOMMANDO**

schrijfwijze

```
STOP [ON  ]
     [OFF ]
     [STOP]
```

betekenis

Met het STOP-kommando kunnen we het programma (tussentijds) beeindigen. Vooral bij het uittesten van een wat groter programma wordt vaak een stop ingelast om tussenresultaten te kunnen bekijken. Voorbeeld:

```
NEW
10 PRINT "REGEL 1"
20 STOP
30 PRINT "REGEL 2"
RUN
REGEL 1
Break in 20
Ok
```

SLEUTELWOORD STR$

soort . . . . **A-FUNKTIE**

schrijfwijze

```
STR$(<N>)
```

betekenis

Deze funktie geeft de tegenovergestelde mogelijkheid van de VAL-funktie. Vanuit een numerieke uitdrukking kan een string worden samengesteld. Bijvoorbeeld:

```
NEW
10 LET A=12.3
20 LET A$=STR$(A)
30 PRINT A$;LEN(A$)
RUN
 12.3 5
Ok
```

Uit dit voorbeeld blijkt dat A$ een voorlopende spatie heeft. STR$ geeft aan positieve waarden altijd een voorloopspatie. Bij negatieve waarden wordt deze spatie door een min-teken vervangen. Bijvoorbeeld:

```
NEW
10 LET A=-12
20 PRINT LEN(STR$(A));"FL. "+STR$(A)
30 STOP
RUN
 3 FL. -12
Break in 30
Ok
```

SLEUTELWOORD # SWAP

soort . . . . **KOMMANDO**

schrijfwijze

```
SWAP<ALFANUMERIEKE VARIABELE>,<ALFANUME-
  RIEKE VARIABELE>|SWAP<NUMERIEKE VARIA-
  BELE>,<NUMERIEKE VARIABELE>
```

betekenis

Met het kommando SWAP kunnen de waarden van twee numerieke of twee alfanumerieke variabelen worden verwisseld. Voorbeeld:

```
NEW
10 A=5:B=4:SWAP A,B:PRINT A;B
RUN
 4  5
Ok
```

```
NEW
10 DIM A(1):A(0)=5:A(1)=123
20 SWAP A(0),A(1)
30 PRINT A(0);A(1)
RUN
 123  5
Ok
```

De tweede in SWAP vermelde variabele moet in het programma reeds eerder zijn toegewezen (een waarde hebben gehad of zijn gedimensioneerd).

SLEUTELWOORD # TAN

soort . . . . **N-FUNKTIE**

schrijfwijze

TAN(<HOEK>)

betekenis

Deze funktie geeft als resultaat de tangens van de waarde van de tussen haakjes vermelde uitdrukking. Deze waarde wordt beschouwd als de uitdrukking van een hoek in radialen.

Alhoewel elke waarde voor de hoek is toegestaan, is het in verband met de precisie van berekenen raadzaam om niet te grote hoekmaten te gebruiken.

```
NEW
10 FOR I=0 TO 90
20 LET HOEK=I/180*3.1415926535
30 PRINT I;":";TAN(HOEK)
40 NEXT I
RUN
```

Een tangenstabel verschijnt op het beeldscherm. Merk op dat in regel 20 de hoek van graden naar radialen geconverteerd wordt. Merk ook op dat, in tegenstelling tot de theorie, TAN geen foutmelding geeft bij een hoek van 90 graden (of $\frac{1}{2}\pi$ radialen), doch slechts een grote waarde geeft.

SLEUTELWOORD TIME

soort .... **SYSTEEMVARIABELE**

schrijfwijze

TIME

betekenis

De MSX-computer bezit een interne klok. Deze klok wordt bij het aanzetten op 0 gezet en elke 1/50 seconde met de waarde 1 verhoogd. De waarden van de systeemklok kunnen we onder de systeemvariabele TIME opvragen en we kunnen de klok ook op 0 zetten.
Bijvoorbeeld:

```
NEW
10 TIME=0
20 FOR I=1 TO 50:LET A=SQR(I):NEXT I
30 LET T=TIME/50
40 PRINT "50 WORTELTREKKINGEN"
50 PRINT "KOSTEN";T;"SECONDEN"
RUN
50 WORTELTREKKINGEN
KOSTEN 6.08 SECONDEN
Ok
```

In bovenstaand voorbeeld werden 50 worteltrekkingen gedaan. Gemeten werd hoeveel seconden hiervoor nodig waren.

Het is niet aan te bevelen om de systeemklok voor nauwkeurige en langdurige tijdmetingen te gebruiken; daarvoor is de systeemklok te onnauwkeurig.

Indien de systeemklok (na ongeveer 22 minuten) de waarde 65535 overschrijdt, begint hij weer bij 0. De systeemklok staat stil wanneer de computer bezig is met niet onderbreekbare acties zoals het lezen van of schrijven naar cassette.

In samenwerking met de RND-funktie kan de TIME-variabele nog een heel dankbare taak vervullen.

Het vervelende van de RND-funktie is, dat een willekeurige, volkomen van de bediening onafhankelijke bepaling van een toevalsgetal niet mogelijk is. Vooral met het programmeren van spelletjes is dit een groot probleem. Door nu op een slimme wijze van TIME gebruik te maken, kan dit probleem voor goed uit de wereld geholpen worden:

```
NEW
10 REM TOEVALSGETALLEN
20 LET A=RND(-TIME)
```

```
30 PRINT RND(1)
40 GOTO 30
RUN
```

Het bovenstaande programma levert steeds een andere reeks toevalsgetallen. Het geheim zit in regel 20 waar (zie de behandeling van RND) steeds weer een andere reeks toevalsgetallen wordt geselekteerd. Doordat TIME 50 maal per seconde verandert is het welhaast onmogelijk voor de mens om via de bediening twee maal dezelfde reeks bewust te selekteren.

N.B.: Het sleutelwoord LET mag niet voor TIME worden gebruikt.

SLEUTELWOORD VAL

---

soort . . . . **FUNKTIE**

schrijfwijze

```
VAL(<A>)
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met deze funktie kan een alfanumerieke uitdrukking numeriek worden onderzocht. De funktie VAL onderzoekt de tussen haakjes opgenomen alfanumerieke uitdrukking van links naar rechts. Zolang geen verboden karakters voor een numerieke konstante worden tegengekomen, blijft deze funktie de alfanumerieke variabele doorwerken. Zodra echter een karakter wordt tegengekomen dat verboden is, stopt de funktie VAL het verdere onderzoek en wordt de tot dat toe opgebouwde numerieke konstante als resultaat gegeven. Bijvoorbeeld:

```
NEW
10 LET A$="123GHJ4"
20 LET A=VAL(A$)
30 PRINT A
```

```
40 STOP
RUN
 123
Ok

10 LET A$="1E22HJHKJ"
RUN
 1E+22
Ok
```

(alleen regel 10 wordt vervangen)

```
NEW
10 LINE INPUT "WAARDE:";A$:LET A=VAL(A$)
20 PRINT "WORTEL=";SQR(A):GOTO 10
RUN
WAARDE:12
WORTEL= 3.4641016151377
WAARDE: (etcetera)
```

SLEUTELWOORD

WIDTH

---

soort . . . . **KOMMANDO**

schrijfwijze

WIDTH<TEKSTREGELBREEDTE>

betekenis

Met het WIDTH-kommando kan de gewenste regelbreedte worden ingesteld. Afhankelijk van de SCREEN-instelling (zie de behandeling van SCREEN) kan de breedte maximaal 32 of 40 karakters zijn. Het minimum is natuurlijk gelijk aan 1. Indien de numerieke uitdrukking die tussen haakjes een gebroken waarde als uitkomst heeft, dan wordt een decimale fractie verwaarloosd. Bij het uitvoeren van een WIDTH wordt het beeldscherm schoongemaakt tenzij de betreffende breedte

reeds geactiveerd was.

Voorbeeld:

```
NEW
10 WIDTH 4
20 PRINT "HALLO ALLEMAAL"
RUN                  (beeldscherm wordt schoongemaakt)
                        HALL
                        O AL
                        LEMA
                        AL
                        Ok
```

9 MSX-FOUTMELDING OP VOLGORDE VAN NUMMER

Hieronder volgen de mogelijke foutmeldingen van MSX-basic. Zij zijn op nummer gesorteerd opgenomen terwijl de betekenis van de foutmelding daar achter is geplaatst.

01 NEXT without FOR	gepoogd werd om een NEXT uit te voeren zonder dat een bijbehorende FOR werd uitgevoerd.
02 Syntax error	er werd een fout in de schrijfwijze ontdekt.
03 RETURN without GOSUB	gepoogd werd om een RETURN uit te voeren terwijl er niet eerder een GOSUB werd uitgevoerd.
04 Out of DATA	gepoogd werd om een READ uit te voeren terwijl er geen DATA meer kon worden gevonden.
05 Illegal function call	er werd gepoogd om een kommando of funktie uit te voeren met niet toegestane waarden.
06 Overflow	het resultaat van een berekening ligt boven het toegestane maximum of onder het toegestane minimum.
07 Out of memory	er werd gepoogd om een kommando uit te voeren dat meer geheugen nodig heeft dan er beschikbaar is.
08 Undefined line number	een niet bestaand programmaregelnummer werd benaderd.
09 Subscript out of range	een array-variabele werd buiten zijn dimensies aangesproken of er werd een verkeerd aantal dimensies genoemd.
10 Redimensioned array	er werd gepoogd om een array-variabele voor een tweede keer te DIMensioneren.
11 Division by zero	gepoogd werd om een deling door nul te doen of een negatieve macht van nul te bepalen.
12 Illegal direct	er werd gepoogd om een kommando

	direkt in te tikken terwijl dit kommando slechts in een programmaregel mag voorkomen.
13 Type mismatch	een numerieke en een alfanumerieke uitdrukking werden met elkaar verwisseld.
14 Out of string space	gepoogd werd om een kommando uit te voeren waardoor het stringgeheugen te klein werd. Vergroot het string-geheugen met CLEAR.
15 String too long	er werd gepoogd om een string samen te stellen die langer werd dan 255 posities.
16 String formula too complex	de alfanumerieke uitdrukking die werd uitgewerkt is te ingewikkeld voor MSX-basic. Splits deze uitdrukking in enkele kleinere.
17 Can't continue	gepoogd werd on een programma met CONT te vervolgen terwijl dat niet (meer) gaat.
18 Undefined user function	gepoogd werd om met FN een niet gedefinieerde funktie aan te roepen.
19 Device I/O error	een fout werd ontdekt tijdens het lezen van/schrijven naar een randapparaat (cassetterecorder of printer e.d.).
20 Verify error	tijdens de controle met CLOAD? werd een verschil ontdekt tussen het programma op cassetteband en het programma in het geheugen.
21 No RESUME	de ERROR-routine werd aangeroepen en een RESUME kon niet worden gevonden.
22 RESUME without error	er werd gepoogd om een RESUME uit te voeren terwijl er geen fout via de ON ERROR GOTO werd gedetecteerd.
24 Missing operand	een noodzakelijke uitdrukking wordt in een kommando niet gevonden.
25 Line buffer overflow	een ingetikte programmaregel

	is te lang voor MSX-basic (langer dan 255 karakters).
50 FIELD overflow	deze foutmelding is wel aanwezig maar kan in het standaard MSX-basic niet voorkomen tenzij met ERROR 50 gegenereerd.
51 Internal error	deze melding wijst op een fout die niet zou mogen kunnen voorkomen. Licht uw computerleverancier of MICROSOFT in.
52 Bad file number	een niet toegestaan kanaalnummer werd gebruikt.
53 File not found	deze foutmelding is wel aanwezig maar kan in het standaard MSX-basic niet voorkomen tenzij met error 53 gegenereerd.
54 File already open	deze foutmelding is wel aanwezig maar kan in het standaard MSX-basic niet voorkomen tenzij met error 54 gegenereerd.
55 Input past end	er werd gepoogd om nog gegevens uit een bestand te lezen terwijl dit bestand reeds volledig was doorgewerkt.
56 Bad file name	een bestandsnaam werd niet volgens de voorschriften samengesteld.
57 Direct statement in file	tijdens het LOADen van een programma werd een direkt kommando (zonder regelnummer) ontdekt; LOAD werd gestopt.
58 Sequential I/O only	deze foutmelding is wel aanwezig maar kan in het standaard MSX-basic niet voorkomen tenzij met ERROR 58 gegenereerd.
59 File not open	een kanaal werd aangesproken zonder dat hierop een bestand geOPENd is.

Indien een niet bestaand foutnummer met ERROR wordt genegeerd, zal de foutmelding 'Unprintable error' verschijnen. Deze foutmeldingen kunnen door de programmeur een bepaalde betekenis worden toegedacht.

				BIT 4,5,6,7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
					0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
					0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
					0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
BIT 0,1,2,3				HEX 1 2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	Null		Spatie	θ	@	P	`	p	ç	É	á	Ã			α	≡
0	0	0	1	1	☺		!	1	A	Q	a	q	Ü	æ	í	ã			β	±
0	0	1	0	2	☻		"	2	B	R	b	r	e	Æ	ó	Ĩ			γ	≥
0	0	1	1	3	♥		#	3	C	S	c	s	à	ô	ú	ĩ			π	≤
0	1	0	0	4	♦		$	4	D	T	d	t	ä	ö	ñ	Õ			Σ	⌠
0	1	0	1	5	♣		%	5	E	U	e	u	à	ò	Ñ	õ			σ	⌡
0	1	1	0	6	♠		&	6	F	V	f	v	å	û	a	Ũ			μ	÷
0	1	1	1	7	•		'	7	G	W	g	w	ç	ù	o	ũ			τ	≈
1	0	0	0	8	◘		(	8	H	X	h	x	e	ÿ	¿	Ĳ		▽	Φ	°
1	0	0	1	9	○		)	9	I	Y	i	y	e	Ö	⌐	ĳ		‡	θ	∙
1	0	1	0	A	◙		*	:	J	Z	j	z	e	Ü	¬	¾		₩	Ω	•
1	0	1	1	B	♂		+	;	K	[	k	{	ï	¢	½	~			δ	√
1	1	0	0	C	♀		,	<	L	\	l	¦	î	£	¼	◊			∞	η
1	1	0	1	D	♪		–	=	M	]	m	}	ì	¥	i	‰			φ	²
1	1	1	0	E	♫		.	>	N	∧	n	~	Ä	Pt	≪	¶			∈	■
1	1	1	1	F	☼		/	?	O	_	o	△	Å	ƒ	≫	§			∩	Transparant

In de bovenstaande tabel zijn alle MSX-karakters opgenomen met hun bijhorende binaire en hexad-cimale waarden.

11 GERESERVEERDE SLEUTELWOORDEN

Deze sleutelwoorden zijn door MSX-basic gereserveerd en mogen niet als vrije variabele-namen worden gebruikt.

ABS	DEFINT	INKEY$	NEXT	SCREEN
AND	DEFDBL	INP	NOT	SET
APPEND	DEFSNG	INPUT	OCT$	SGN
AS	DEFSTR	INPUT$	OFF	SIN
ASC	DEFUSR	INSTR	ON	SOUND
ATN	DELETE	INT	OPEN	SPACE$
ATTR$	DIM	INTERVAL	OR	SPC
AUTO	DRAW	KEY	OUT	SPRITE
BASE	DSKF	KILL	OUTPUT	SPRITE$
BEEP	DSKI$	LEFT$	PAD	SQR
BIN$	DSKO$	LEN	PAINT	STEP
BLOAD	ELSE	LET	PDL	STICK
BSAVE	END	LFILES	PEEK	STOP
CALL	EOF	LINE	PLAY	STR$
CDBL	EQV	LIST	POINT	STRIG
CHR$	ERASE	LLIST	POKE	STRING$
CINT	ERL	LOAD	PORT	SWAP
CIRCLE	ERR	LOC	POS	TAB
CLEAR	ERROR	LOCATE	PRESET	TAN
CLOAD	EXP	LOF	PRINT	THEN
CLOSE	FIELD	LOG	PSET	TIME
CLS	FILES	LPOS	PUT	TO
COLOR	FIX	LPRINT	RANDOM	TROFF
CONT	FN	LSET	READ	TRON
COPY	FOR	MAXFILES	REM	USING
COS	FPOS	MERGE	RENUM	USR
CSAVE	FRE	MID$	RESTORE	VAL
CSNG	GET	MKD$	RESUME	VARPTR
CSRLN	GOSUB	MKI$	RETURN	VDP
CVD	GOTO	MOD	RIGHT$	VPEEK
CVI	GO TO	MOTOR	RND	VPOKE
CVS	HEX$	MKS$	RSET	WAIT
DATA	IF	NAME	RUN	WIDTH
DEF	IMP	NEW	SAVE	XOR

ENKELE MSX-uitgaven

serie: UW MSX COMPUTER DE BAAS

MSX BASIC HANDBOEK voor iedereen, door A.C.J. Groeneveld
Hèt handboek voor iedere MSX computer gebruiker
ISBN 90 6398 100 7

MSX DISK HANDBOEK voor iedereen, door A.C.J. Groeneveld
ISBN 90 6398 407 3

MSX ZAKBOEKJE voor iedereen, door Wessel Akkermans
Alle belangrijke gegevens voor zowel de BASIC- als machinetaal-programmeurs, voor zover mogelijk in de vorm van overzichten en tabellen. ISBN 90 6398 888 5

MSX BASIC leerboek deel 1, door Wessel Akkermans en Piet den Heijer. Informatie over MSX-hardware en leren programmeren
ISBN 90 6398 649 1

MSX BASIC leerboek deel 2, door Wessel Akkermans en Piet den Heijer. Doorgaan met leren programmeren en speciale aandacht voor kleur, grafieken, geluid/muziek en joy-sticks
ISBN 90 6398 769 2

MSX DOS leerboek, door Wessel Akkermans en Piet den Heijer. Het manipuleren met strings, het bewerken en opslaan van gegevens op cassette en schijf
ISBN 90 6398 519 3

SOFTWARE PLUS in MSX

INTROTAPE MSX, door A.C.J. Groeneveld. Begeleid met instrukties om de computer aan te sluiten en de tape te laden, wordt MSX op een vriendelijke en onderwijzende manier vanuit nul bij de gebruiker geïntroduceerd. Na het doorwerken van deze software is de gebruiker zelf in staat MSX-basic programma's te schrijven
ISBN 90 6398 148 1

MSX-SCRIPT door Ton Weijters
Een menu-gestuurde nederlandstalige tekstverwerker
ISBN 90 6398 189 9

AVT electronics

AVT electronics bv.,
Postbus 61411, 2506 AK Den Haag